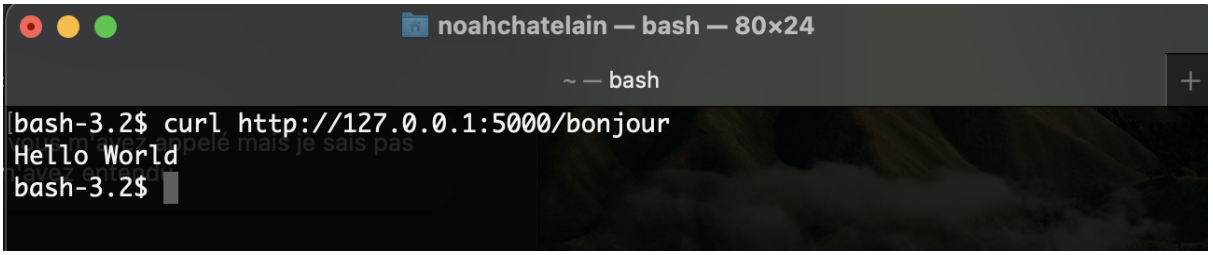
TP1 WebServers - Noah Chatelain & Ambroise Gyre

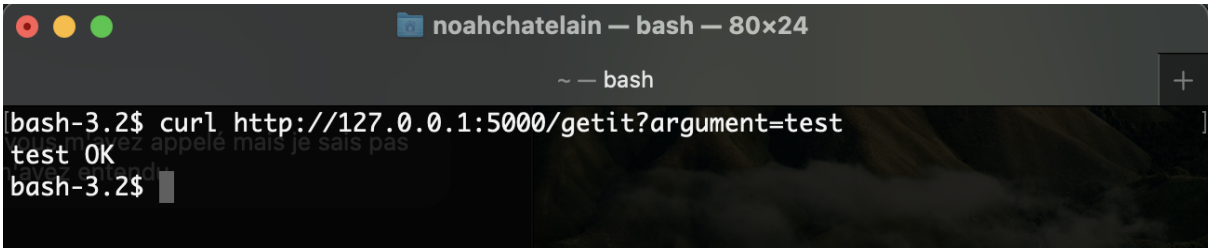
⚠️ Pour vos tests, le mot de passe par défaut de tous les étudiants est : **123456**
Il apparaît dans la base de donné hashé en SHA256

```
MBP-de-Noah:TP Webserver noahchatelain$ python3 app.py
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

http://127.0.0.1/5000/bonjour



http://127.0.0.1/5000/getit?argument=test

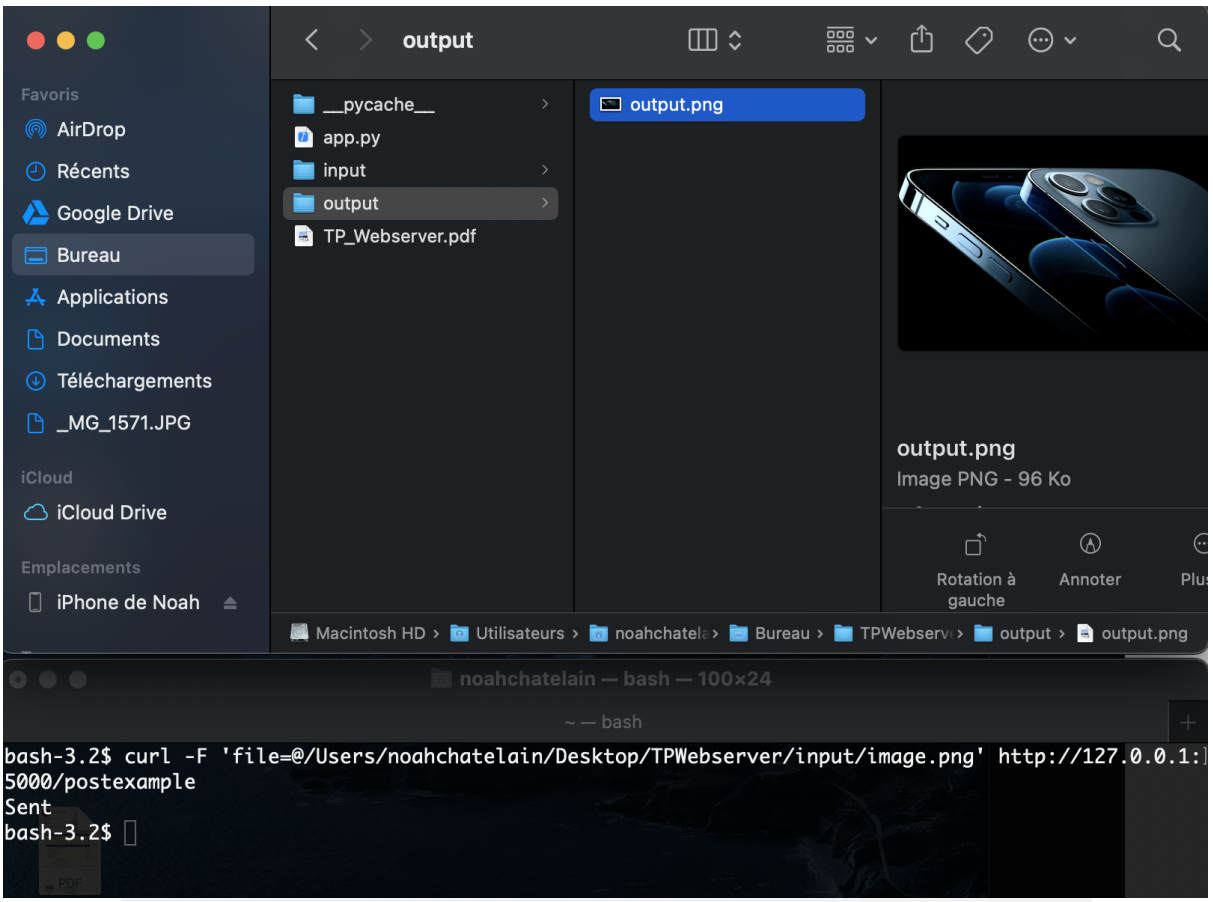


curl 127.0.0.1:5000/getit?arga=test1&argb=test2
(sans modification permettant l'addition de deux paramètres)

curl 127.0.0.1:5000/addition/10/14
(avec modification permettant l'addition de deux paramètres)

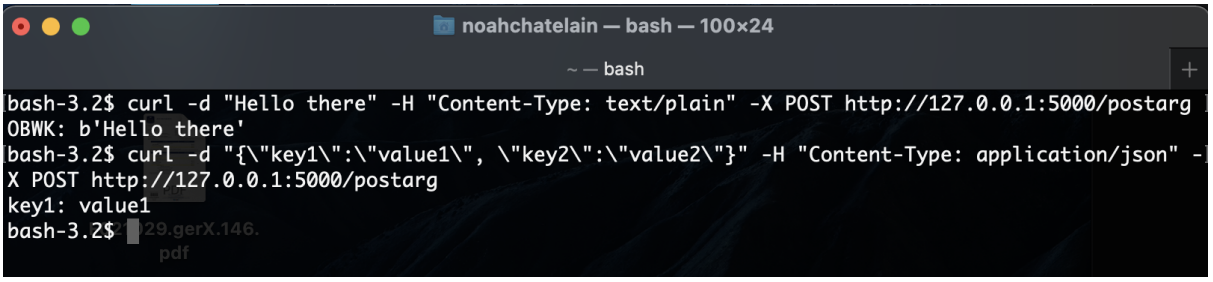
http://127.0.0.1:5000/compare/test1/test2
(ajout de la route permettant la comparaison de deux chaînes de caractères)

Réception d'un fichier et requête POST :

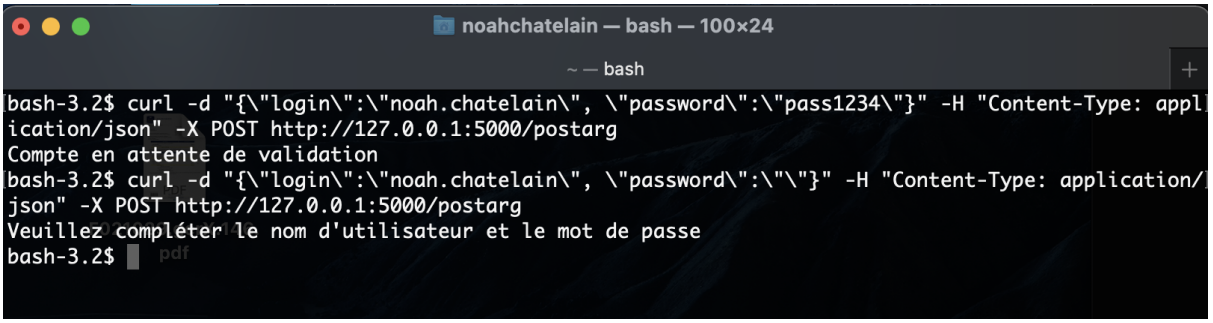



```python
@app.route('/postexample', methods=['POST'])
def postexample():
    if request.method == 'POST':
        if 'file' not in request.files:
            flash('No file part')
            return redirect(request.url)
        file = request.files['file']
        if file.filename == '':
            flash('No selected file')
            return redirect(request.url)
        file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))
    return "Sent\n"
```

Traitement différencié selon le type de contenu (texte pur ou json).
Explication demandée :
Les 2 requêtes *curl* du TP permettent de passer dans le Header de la requête POST le Content-Type de ce que l'on va envoyer. Ainsi, dans notre API, nous pourrons faire des conditions sur ce Content-Type et effectuer des actions différentes.



Accepter des informations de login et mot de passe dans un json, pour l'instant toujours répondre « Compte en attente de validation » si un json avec des informations de login ET de mot de passe non vides sont présentes.



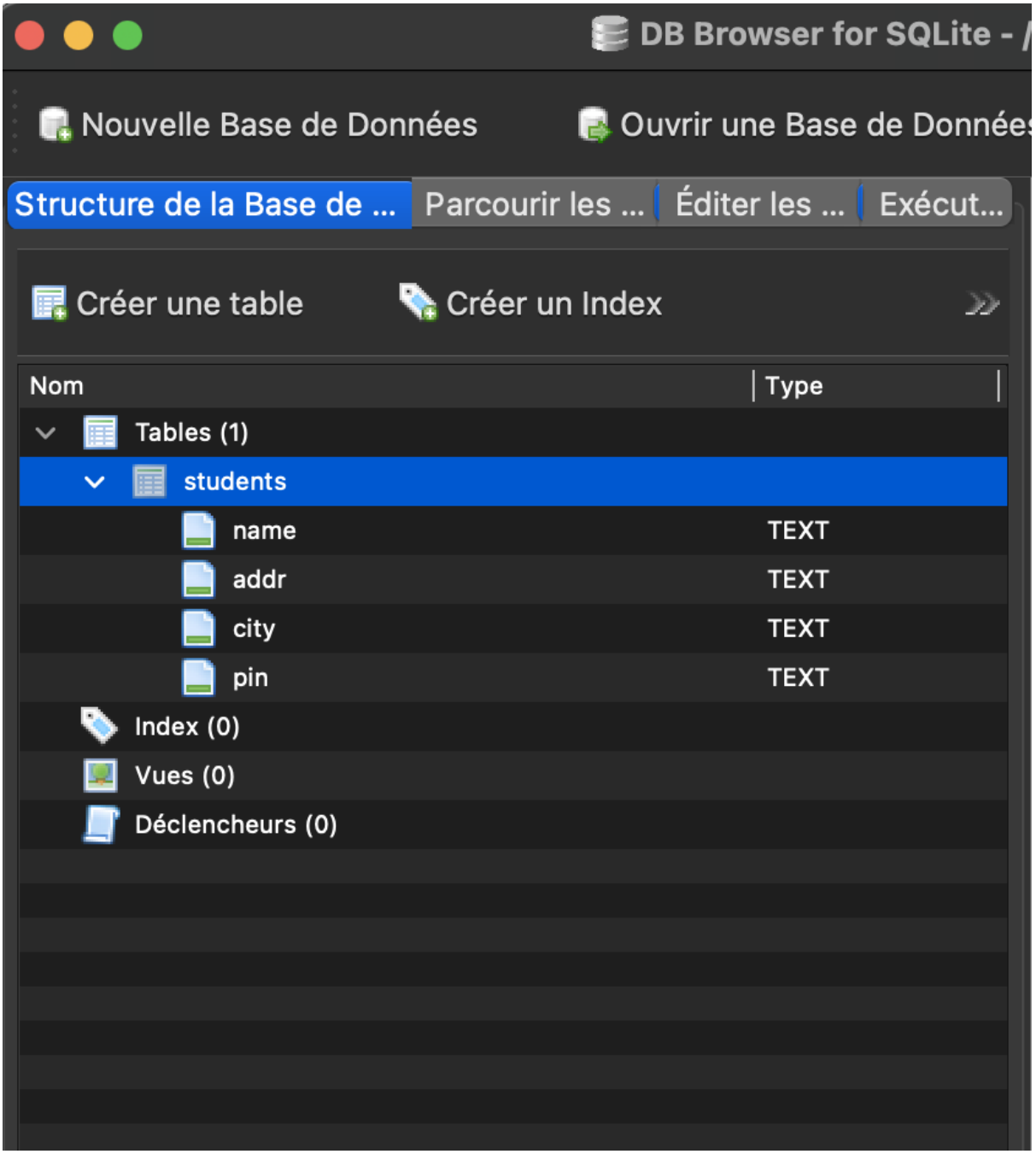

```python
@app.route('/postarg', methods=['POST'])
def postarg():
    if request.headers['Content-Type'] == 'text/plain':
        return "OBWK: " + str(request.data) + "\n"
    elif request.headers['Content-Type'] == 'application/json':
        if request.json["login"] != "" and request.json["password"] != "":
            return "Compte en attente de validation\n"
        else:
            return "Veuillez compléter le nom d'utilisateur et le mot de passe\n"
    return "NOK"
```

Création d'une nouvelle route qui permettra la création d'une base de données :

```python
@app.route('/createmanu')
def createmanu():
    conn = sqlite3.connect('database.db')
    print ("Opened database successfully")
    #type : https://www.sqlite.org/datatype3.html
    conn.execute('CREATE TABLE students (name TEXT, addr TEXT, city TEXT, pin TEXT)')
    print("Table created successfully")
    conn.close()
    return 'Your table is created.'
```

```
MBP-de-Noah:TPWebserver noahchatelain$ python3 app.py
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
Opened database successfully
Table created successfully
127.0.0.1 - - [23/Sep/2021 09:16:02] "GET /createmanu HTTP/1.1" 200 -
```

Your table is created.



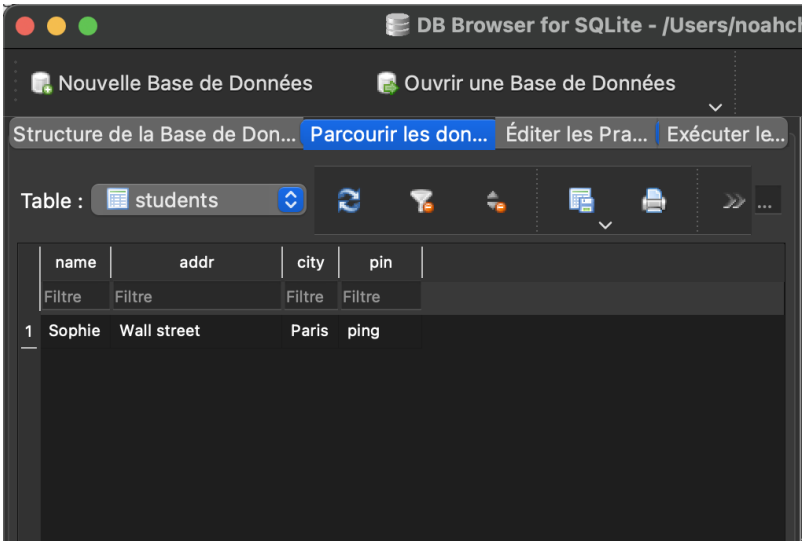
Résultat visuel de la création de la table dans DB Browser for SQLite

/addmanu

```python
@app.route('/addmanu')
def addmanu():
    conn = sqlite3.connect('database.db')
    cursor = conn.cursor()
    oneRecord = ['Sophie','Boulevard Concorde','Paris','Xae-A12']
    cursor.execute('INSERT INTO students VALUES (?,?,?,?)', oneRecord)
    conn.commit()
    conn.close()
    return 'OK'
```

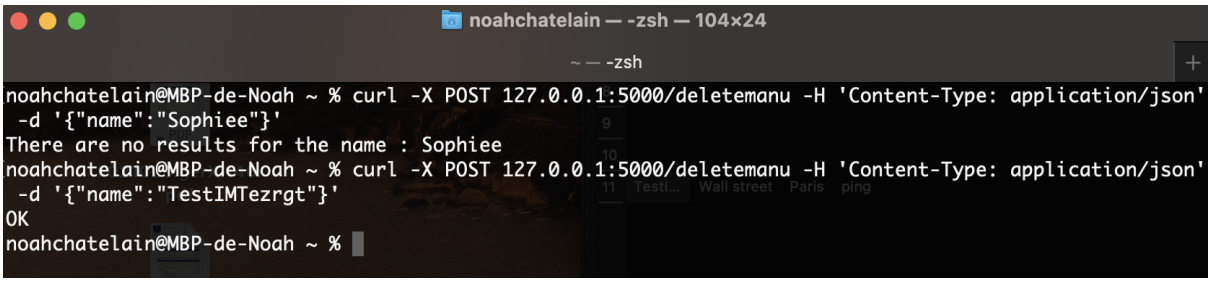
Modifier la méthode en requête POST :

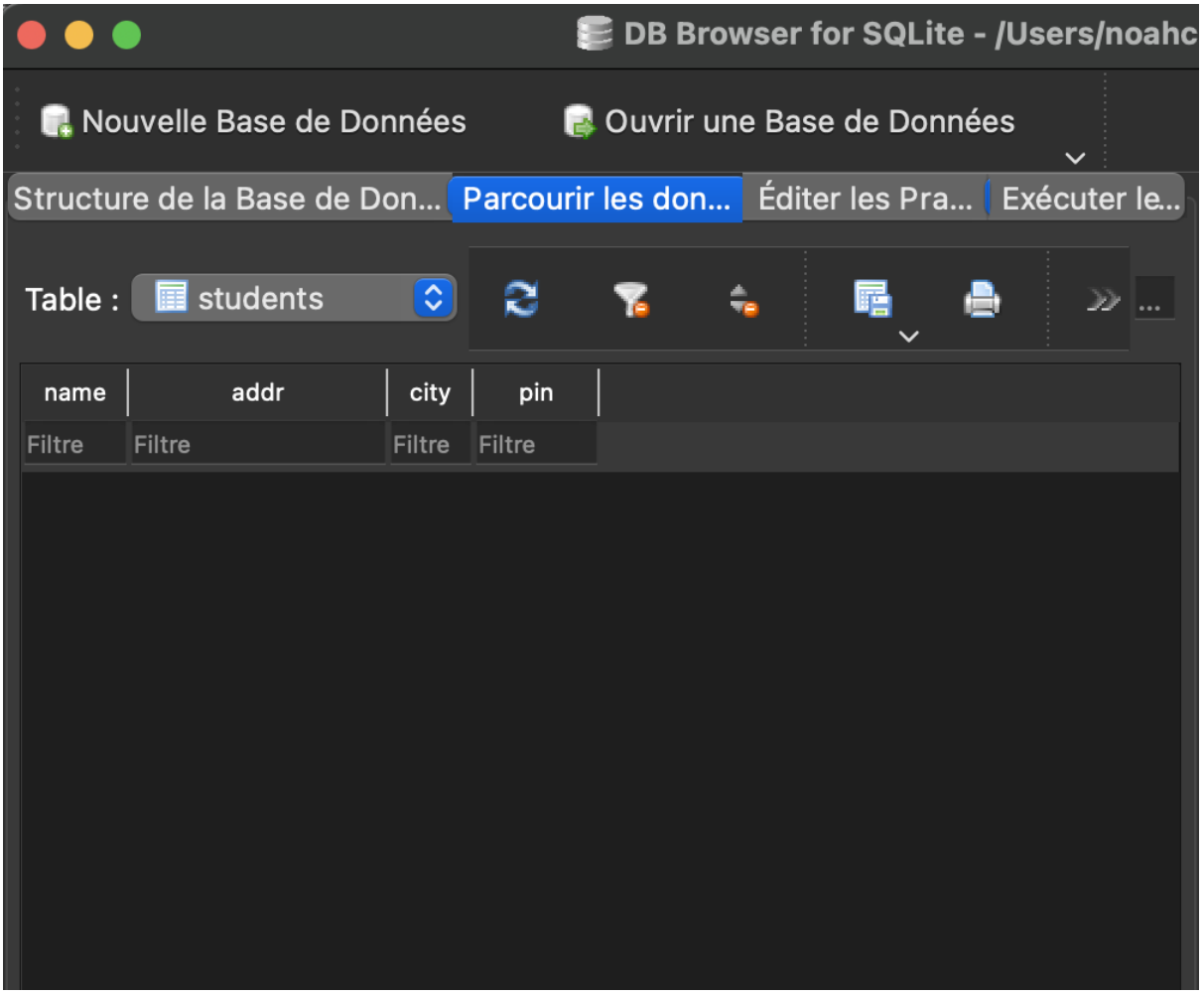
```python
@app.route('/addmanu', methods=['POST'])
def addmanu():
    if request.headers['content-type'] == 'application/json':
        name = request.json["name"]
        addr = request.json["addr"]
        city = request.json["city"]
        pin = request.json["pin"]
        conn = sqlite3.connect('database.db')
        cursor = conn.cursor()
        oneRecord = [name, addr, city, pin]
        cursor.execute('INSERT INTO students VALUES (?,?,?,?)', oneRecord)
        conn.commit()
        conn.close()
        return 'OK\n'
    return "NOK"
```



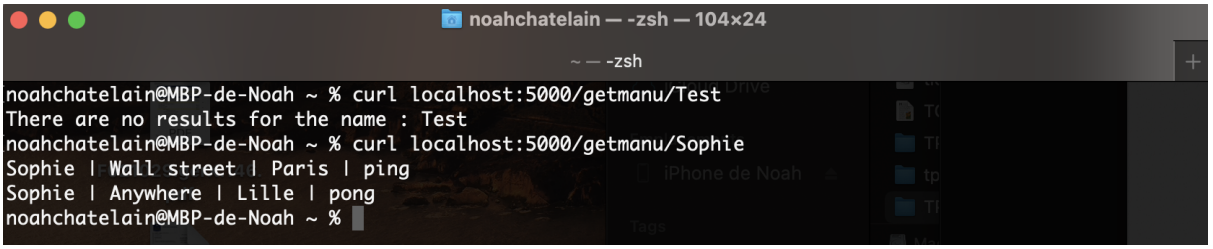
/deletemanu

```python
@app.route('/deletemanu', methods=['POST'])
def deletemanu():
    if request.headers['content-type'] == 'application/json':
        if request.json["name"] != "":
            name = request.json["name"]
            if checkIfNameExistInDb(name):
                conn = sqlite3.connect('database.db')
                cursor = conn.cursor()
                oneRecord = [name]
                cursor.execute('DELETE FROM students WHERE name = ?', oneRecord)
                conn.commit()
                conn.close()
                return 'OK\n'
            else:
                return str("There are no results for the name : " + name + "\n")
    return "NOK"
```

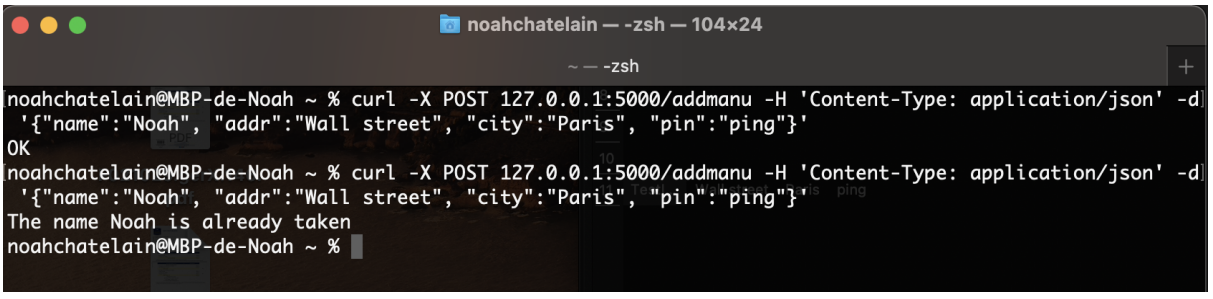




Récupération des informations d'un étudiant : /getmanu/<student_name>



Modification de la première route : Vérifier s'il n'y a pas de doublons à l'ajout



Création de la fonction *nameInTheDataBase()* pour vérifier si *name* existe dans la BDD

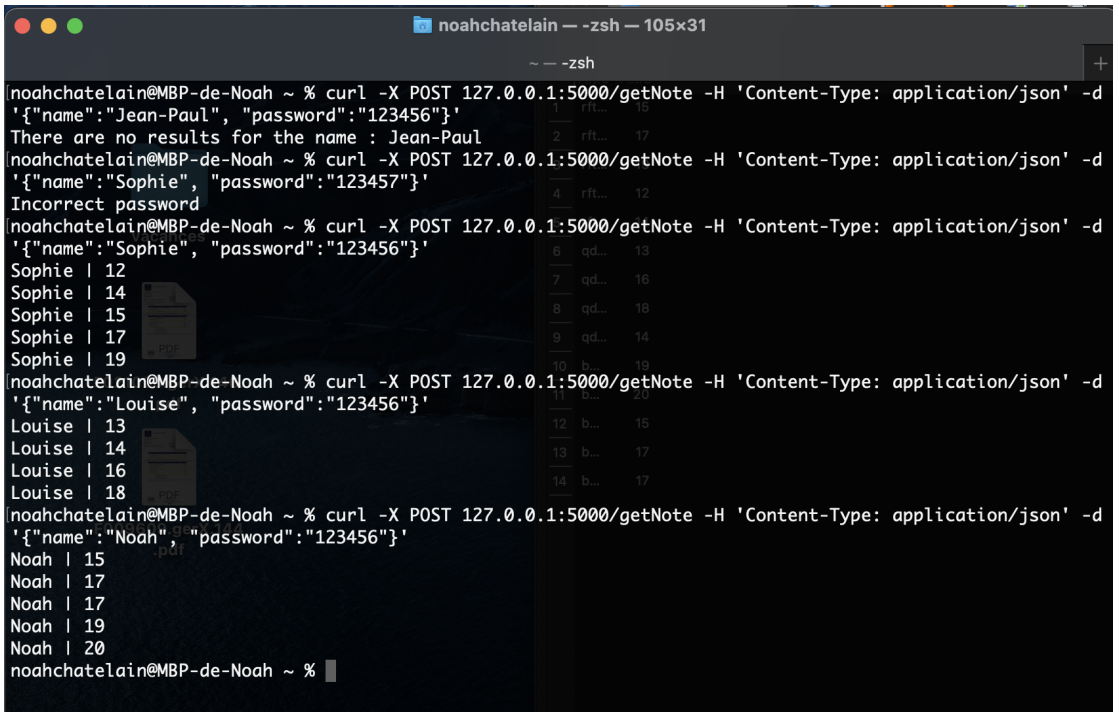
```python
def nameInTheDataBase(name):
    conn = sqlite3.connect('database.db')
    cursor = conn.cursor()
    oneRecord = [name]
    cursor.execute("SELECT * FROM students WHERE name = ?", oneRecord)
    rows = cursor.fetchall()
    return len(rows) >= 1
```

Ajout de la mise à jour d'un étudiant avec la méthode HTTP PUT :

```python
@app.route('/updatemanu', methods=['PUT'])
def updatemanu():
    if request.headers['content-type'] == 'application/json':
        name = request.json["name"]
        addr = request.json["addr"]
        city = request.json["city"]
        pin = request.json["pin"]
        if nameInTheDataBase(name):
            conn = sqlite3.connect('database.db')
            cursor = conn.cursor()
            oneRecord = [addr, city, pin, name]
            cursor.execute('UPDATE students SET addr = ?, city = ?, pin = ? WHERE name = ?', oneRecord)
            conn.commit()
            conn.close()
            return 'OK\n'
    return str("The name " + name + " is already taken\n")
```

Ajout de la table *notes* qui relie l'ID de la table à la valeur *pin* de la table *students*.
Grâce à ça, on peut désormais retrouver toutes les notes d'un étudiant avec une requête POST avec en body le JSON du *name* et du *password* :
curl -X POST 127.0.0.1:5000/getNote -H 'Content-Type: application/json' -d '{"name":"Sophie", "password":"123456"}'
Avec les messages d'erreurs demandés, tels que "*Incorrect password*" ou encore "*There are no results for the name xxxx*"

```python
@app.route('/getNote', methods=['POST'])
def getNote():
    if request.headers['content-type'] == 'application/json':
        if request.json["name"] != "" and request.json["password"] != "":
            name = request.json["name"]
            password = request.json["password"]
            if nameInTheDataBase(name):
                if nameAndPasswordExist(name, password):
                    conn = sqlite3.connect('database.db')
                    cursor = conn.cursor()
                    oneRecord = [name, password]
                    cursor.execute('SELECT name, note FROM students INNER JOIN notes ON students.pin = notes.id WHERE students.name = ? and students.password = ?', oneRecord)
                    rows = cursor.fetchall()
                    result = ""
                    for row in rows:
                        result += str(row[0] + " | " + str(row[1])) + "\n"
                        print(row)
                    return str(result)
                else:
                    return str("Incorrect password\n")
            else:
                return str("There are no results for the name : " + name + "\n")
    return "NOK"
```



Résultat des erreurs de *name* inexistants et de mot de passe incorrects, et des notes

En revoyant le JSON avec uniquement les notes de l'étudiant :

```python
@app.route('/getNote', methods=['POST'])
def getNote():
    if request.headers['content-type'] == 'application/json':
        if request.json["name"] != "" and request.json["password"] != "":
            name = request.json["name"]
            password = request.json["password"]
            if nameInTheDataBase(name):
                if nameAndPasswordExist(name, password):
                    conn = sqlite3.connect('database.db')
                    cursor = conn.cursor()
                    oneRecord = [name, password]
                    cursor.execute('SELECT name, note FROM students INNER JOIN notes ON students.pin = notes.id WHERE students.name = ? and students.password = ?', oneRecord)
                    rows = cursor.fetchall()
                    result = ""
                    for row in rows:
                        result += '{"note":' + str(row[1]) + '},'
                        print(row)
                    result = "[" + result + "]"
                    return jsonify(result)
                else:
                    return str("Incorrect password\n")
            else:
                return str("There are no results for the name : " + name + "\n")
    return "NOK"
```

```
noahchatelain — -zsh — 105×31
~ — -zsh
noahchatelain@MBP-de-Noah ~ % curl -X POST 127.0.0.1:5000/getNote -H 'Content-Type: application/json' -d
'{"name":"Noah", "password":"123456"}'
"[{\"note\":15},{\"note\":17},{\"note\":17},{\"note\":19},{\"note\":20},]"
noahchatelain@MBP-de-Noah ~ %
```

L'utilisateur dispose d'un lien d'inscription /subscribe qui lui permet de compléter ses informations avec un mot de passe, et son identifiant unique qui le relie à la table *note* est généré aléatoirement via une chaîne de caractères de taille 6.

```python
@app.route('/subscribe', methods=['POST'])
def subscribe():
    if request.headers['content-type'] == 'application/json':
        name = request.json["name"]
        addr = request.json["addr"]
        city = request.json["city"]
        password = request.json["password"]
        if not nameInTheDataBase(name):
            conn = sqlite3.connect('database.db')
            cursor = conn.cursor()
            oneRecord = [name, addr, city, createRandomString(), password]
            cursor.execute('INSERT INTO students VALUES (?,?,?,?,?)', oneRecord)
            conn.commit()
            conn.close()
            return 'OK\n'
    return str("The name " + name + " is already taken\n")
```

```
noahchatelain — -zsh — 105×31
~ — -zsh
noahchatelain@MBP-de-Noah ~ % curl -X POST 127.0.0.1:5000/subscribe -H 'Content-Type: application/json' -
d '{"name":"NewStudent","addr":"IMT","city":"Lille","password":"123456"}'
OK
noahchatelain@MBP-de-Noah ~ %
```

Création d'une fonction *stringToSha256()* afin de n'avoir aucun mot de passe en clair dans la base de données.

```python
def stringToSha256(string):
    return hashlib.sha256(string.encode()).hexdigest()
```

```python
@app.route('/getNote', methods=['POST'])
def getNote():
    if request.headers['content-type'] == 'application/json':
        if request.json["name"] != "" and request.json["password"] != "":
            name = request.json["name"]
            password = stringToSha256(request.json["password"])
            if nameInTheDataBase(name):
                if nameAndPasswordExist(name, password):
                    conn = sqlite3.connect('database.db')
                    cursor = conn.cursor()
                    oneRecord = [name, password]
                    cursor.execute('SELECT name, note FROM students INNER JOIN notes ON students.pin = notes.id WHERE students.name = ? and students.password = ?', oneRecord)
                    rows = cursor.fetchall()
                    result = ""
                    for row in rows:
                        result += '{"note":' + str(row[1]) + '},'
                    result = "[" + result + "]"
                    return jsonify(result)
                else:
                    return str("Incorrect password\n")
            else:
                return str("There are no results for the name : " + name + "\n")
    return "NOK"
```

Ajout de la possibilité de mettre à jour la photo d'un étudiant, la nouvelle photo envoyée est stockée dans le dossier *students_photo* :

curl -X POST -i -F "name=Noah" -F "password=123456" -F "file=@/path/to/image.png" 127.0.0.1:5000/updatePhoto