SQL et Données structurées

Sami Souihi

v1_2020

Définition

SQL (Structured Query Language) est un langage qui permet de communiquer avec les bases de données. Il permet créer des tables, de rechercher, d’ajouter, de modifier ou de supprimer des données dans la BD.

Objectifs

DÉCOUVERTE DE SQLITE

INTÉGRATION SQLITE & FLASK

OBJECT-RELATIONAL MAPPING (ORM)

SQLite

- SQLite est une bibliothèque écrite en langage C qui propose un moteur de base de données relationnelle accessible par le langage SQL
- Une base SQLite3 a la particularité d'être contenue dans un fichier qui porte le même nom. Le moteur de base de données SQLite3 est une bibliothèque, libsqlite3, qu'il est possible d'utiliser avec l'interface utilisateur en ligne de commande sqlite3 ou, via une API développée pour un langage donnée .

Créer votre première base de données et table

- La base de données SQLite stocke toutes les données dans un seul fichier. Le programme ci-dessous crée une base de données SQLite « database.db » ainsi qu'une table « etudiants ».

```
import sqlite3

conn = sqlite3.connect ( 'database.db' )
print ("Base de données ouverte avec succès")

conn.execute ( 'CREATE TABLE etudiants (nom
TEXT, addr TEXT, pin TEXT)' )
print ("Table créée avec succès")
conn.close ()
```

Insérer un étudiant

```
with sqlite3.connect("database.db") as
con:
        cur = con.cursor()
        cur.execute("INSERT INTO etudiants
(nom,addr,pin) VALUES (?,?,?)", (" John
Doe" ," 122 rue paul armangot" ," 123"))
        con.commit()
con.close()
```

Intégration dans flask

New.html

```
<html>
<body>
<form action =    " http://127.0.0.1:5000/new " method = "POST">

<h3>Nouveau étudiant </h3>

Nom<br> <input type = "text" nom = "n" /></br>
Adresse<br> <textarea addr = "add" ></textarea><br>
PINCODE<br> <input type = "text" name = "pin" /><br>

<input type = "submit" value = "Envoyer" /><br>

</form>

</body>
</html>
```

Réaliser le code du contrôleur correspondant

Exercice

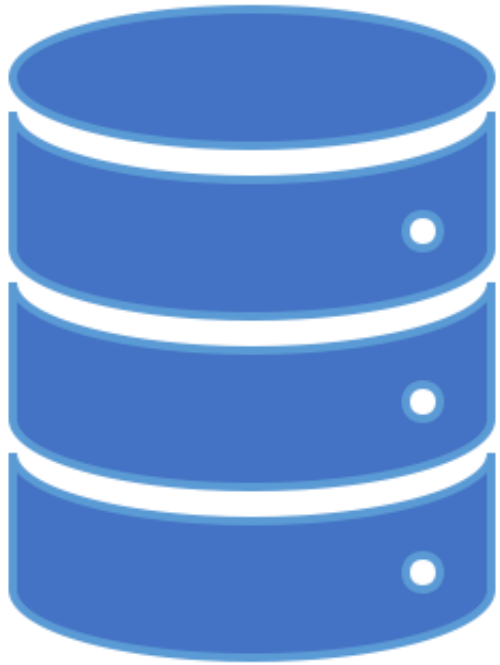
Créer une application qui permet d'enregistrer une liste d'étudiants et de l'afficher. Chaque étudiant aura pour paramètre un nom, une adresse, et un pincode.
L'adresse et le pincode pourront être modifiés.

>> La liste des étudiants doit être enregistrée dans une base de données gérées par sqlite3
>> l'application doit adopter une strucutre MVC

>> ajouter une route permettant d'authentifier l'admin avec jwt

Object-Relational Mapping (ORM)

Écrivez Python au lieu de SQL!

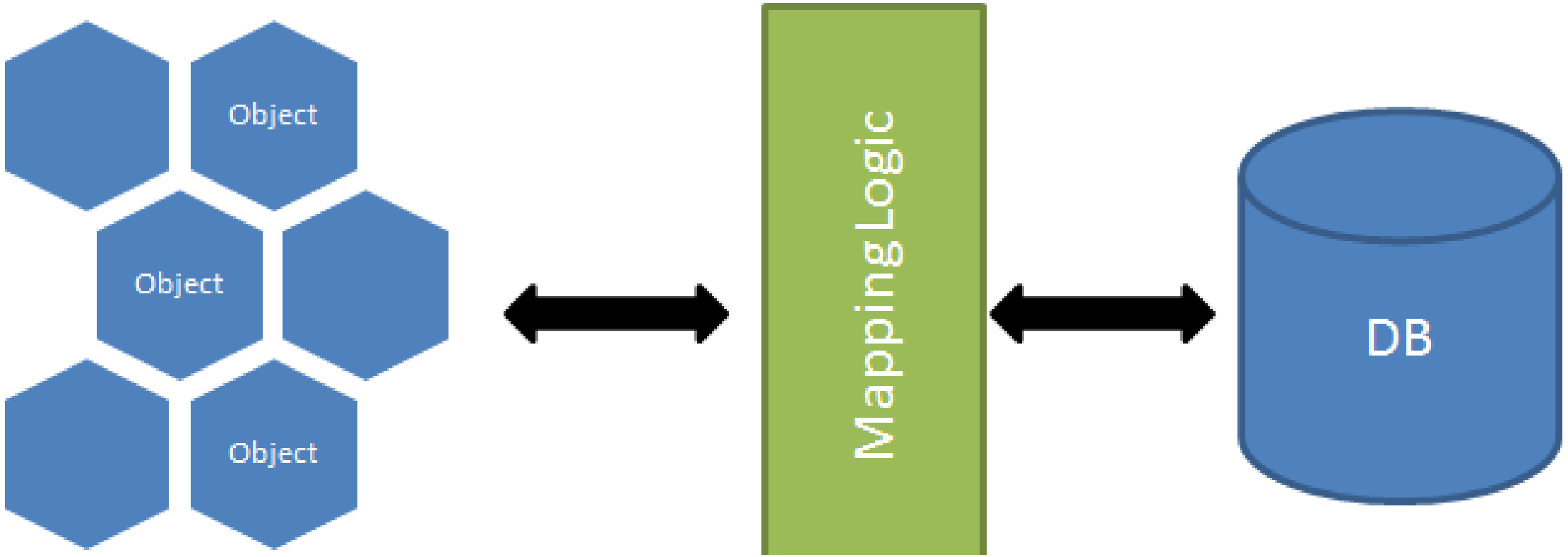


Objectif principal - objets persistants

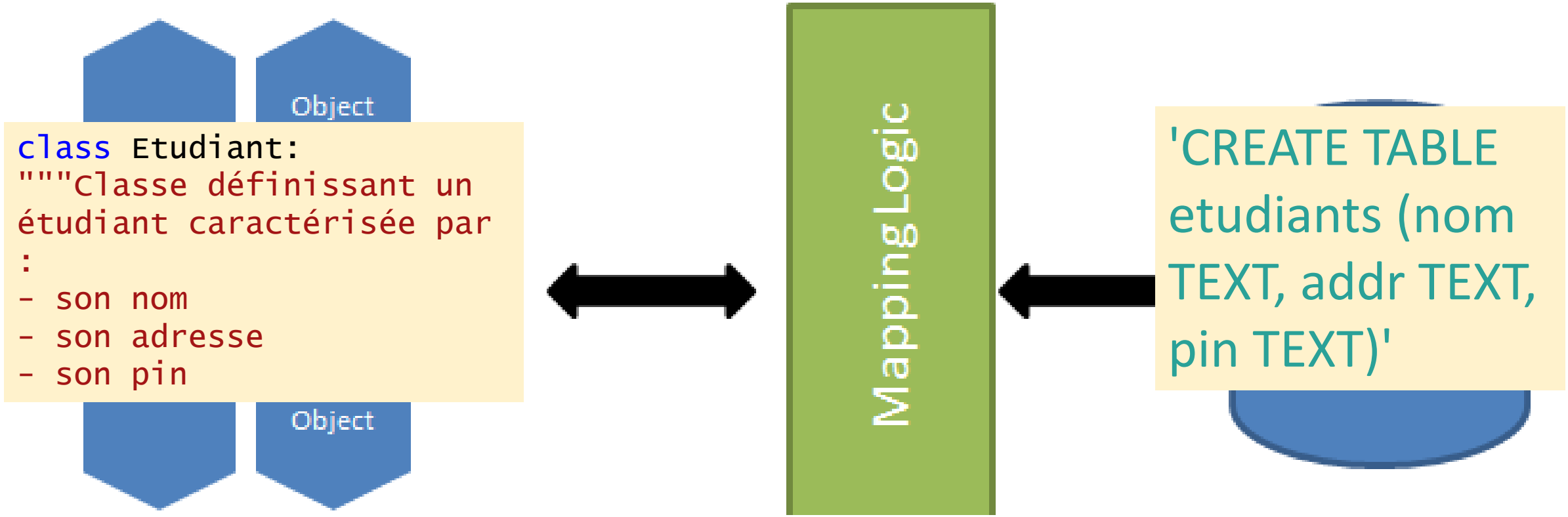
La BD est vue comme un outil pour implémenter des objets persistants.

Le programmeur pense en termes d'application et d'objets.

Spécifiez les objets persistants => l’Interaction avec la BD est Masquée



Focus sur les données



Focus sur les données

SQLAlchemy

SQLAlchemy est une boîte à outils pour base de données largement utilisée. Contrairement à de nombreuses bibliothèques de base de données, il ne fournit pas seulement une couche ORM, mais aussi une API généralisée pour l'écriture du code agnostique aux bases de données, sans SQL.

SQLAlchemy & Flask -Minimal

```
from flask import Flask
from flask_sqlalchemy import SQLAlchemy

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:////tmp/test.db'
db = SQLAlchemy(app)


class Etudiant(db.Model):
        id = db.Column(db.Integer, primary_key=True)
        nom = db.Column(db.String(80), unique=True, nullable=False)
        adresse = db.Column(db.String(120), unique=True, nullable=False)
        pin = db.Column(db.String(20), unique=True, nullable=False)


def __repr__(self):
        return '<Etudiant %r>' % self.username
```

Les commandes de base

- db.create_all**() :**
- john = Etudiant**(**nom=john'**,** adresse=122 rue paul armangot', pin='123 »**)**
- db.session.add**(**john**)**
- db.session.commit**()**
- Etudiant.query.all**()**
- Etudiant.query.filter_by**(**nom='john'**)**.first**()**

Allons plus loin : Les relations

Exercice: Créer une Class : Groupe et instancier un objet de cette classe que vous nomerez ITS2 et qui contient 3 étudiants que vous aurez créer.

Dans la classe « Etudiant »

- group_id = db.Column(db.Integer, db.ForeignKey(‘group.id’)

Dans la classe « Group »

- etudiant = db.relationship(‘Etudiant', backref=etudiant’)

Vous venez de créer une relation un à plusieurs. Il est aussi possible de créer d’autre type de relations

PS: Vous pouvez vous appuyer sur la documentation officielle de flask : https://flask-sqlalchemy.palletsprojects.com/en/2.x/quickstart/

Exercice:

(1) Faire tourner l'app de gestion d'étudiants avec Alchemy

(2) Ajouter une authentification avec token jwt

Jwt: https://jwt.io/introduction

Un exemple de tuto d'intégration: https://www.geeksforgeeks.org/using-jwt-for-user-authentication-in-flask/

(3) Ajouter une documentation avec SwaggerUI

https://github.com/swagger-api/swagger-ui

https://medium.com/the-ai-analytics-corner/swagger-ui-to-your-python-flask-api-9bf1c8fc0178

(4) Faire tourner une api sur docker > dockeriser l'api « webApi » du début du cours

Piste: https://blog.jaaj.dev/2023/02/10/Comment-dockeriser-une-application-flask.html

Ressources d'images docker: https://hub.docker.com/