Université de Yaoundé I
Faculté des sciences
Département d'informatique

University of Yaounde I
Faculty of sciences
Computer sciences department

RAPPORT TP INF222

TCHANA YOTAT WILFRIED TREVOR

MATRICULE : 23U2551

Examinateur : Dr Jiomekong

1. Introduction

Dans le cadre du TP de l’unité d’enseignement INF222 , il nous a été demandé de concevoir une application de gestion de l’alimentation des utilisateurs. De ce fait, ce rapport a pour but de présenter l’application conçue. Cette application est une API construite avec Flask, permettant de gérer des interactions avec une base de données PostgreSQL via le module psycopg2. L'application permet d'ajouter des données sur des personnes, des nourritures et des ingrédients, tout en offrant des opérations de base sur la base de données.

2. Architecture de l'Application

2.1. Technologies Utilisées

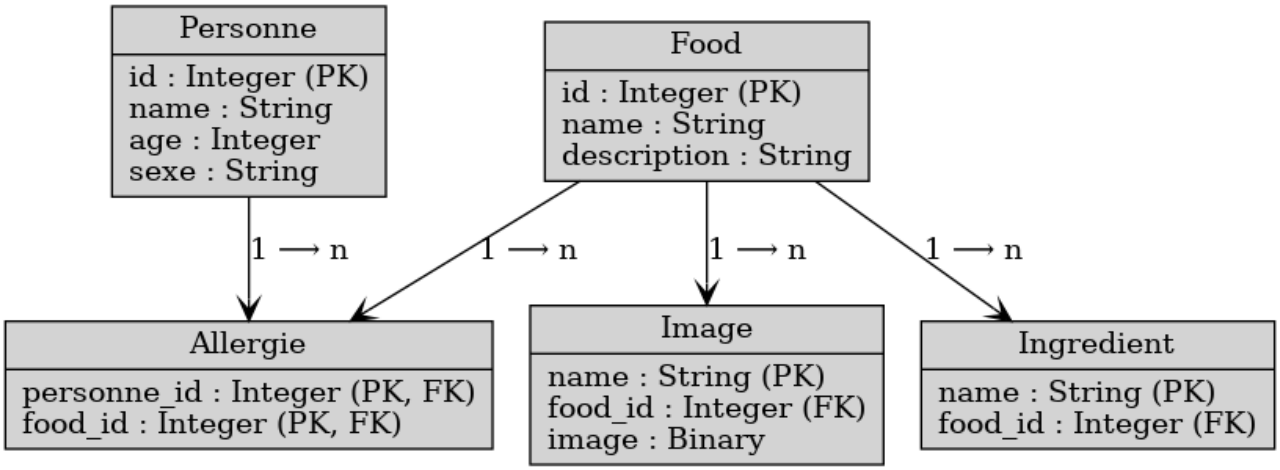
- **Langage de Programmation: Python**
- **Framework: Flask**
- **Base de Données: PostgreSQL**
- **Connecteur: psycopg2**
- **Containerisation: Docker**
- **API : GEMINI (google-genai)**

2.2. Fonctionnalités

- Ajouter une personne.
- Ajouter une nourriture.
- Ajouter des ingrédients.
- Relier une personne à ce qu'elle a mangé.
- Discuter avec un chat bot

3. Diagramme de Classe

Voici un diagramme de classe représentant les principales entités de l'application :



3.1. Relations

- Personne a une relation avec Manger (une personne peut manger plusieurs nourritures).
- Food a une relation avec Ingredient (une nourriture peut avoir plusieurs ingrédients).
- Manger relie Personne et Food.

3.2. Conception du chat bot

Notre chat bot est accessible par la route /poser-question , il a été réalisé à partir de l' API de Gemini. Nous avons ainsi obtenu une clé d' API qui nous a servi à la réalisation de ce chat bot. La route est basé sur une fonction poser_question qui récupère la question de l'utilisateur saisie dans un formulaire et la passe en paramètre à la fonction obtenir_reponse_gemini qui elle à son tour envoi une requête à gemini puis renvoi une réponse qui sera ainsi retournée à l'utilisateur.

Extrait de la fonction obtenir_reponse_gemini :

```
def obtenir_reponse_gemini(question):
        try:
        chat = client.chats.create(model='gemini-2.0-flash')
        response = chat.send_message(question)

        # Extraction de la réponse
        if response.candidates and len(response.candidates) > 0:
        # Accéder au premier candidat
        first_candidate = response.candidates[0]
        # Accéder au contenu
        content = first_candidate.content
        # Extraire le texte
        text_response = content.parts[0].text
        return text_response
        else:
        return "Aucune réponse trouvée."

        except Exception as e:
        print(f"Erreur lors de l'obtention de la réponse : {e}")
        return "Erreur lors de la génération de la réponse"
```

4. Schéma d'Exécution via Docker

4.1. Structure du Dockerfile

Le fichier Dockerfile est structuré pour installer les prérequis nécessaires et l'application.

```
# Utiliser une image de base Python
FROM python:3.10-slim

# Installer les dépendances système
RUN apt-get update && apt-get install -y \
gcc \
libpq-dev \
python3-dev \
&& rm -rf /var/lib/apt/lists/*

# Définir le répertoire de travail
WORKDIR /FOODAPP

# Copier les fichiers de votre projet dans le conteneur
COPY requirements.txt requirements.txt
RUN pip install --upgrade pip && pip install --no-cache-dir -r requirements.txt
COPY . .

# Exposer le port de l'application
EXPOSE 5000
# Commande pour exécuter l'application
CMD ["flask", "run", "--host=0.0.0.0"]
```

4.2. Workflow Docker

1. Construction de l'Image

- L'image Docker est construite à l'aide de la commande suivante : bash

```
•  docker build -t mon_application_food .
```

- **Exécution du Conteneur**

Une fois l'image construite, le conteneur est exécuté : bash

```
docker run -p 5000:5000 mon_application_food
```

2. Déploiement de l'image

Le déploiement quant à lui s'est fait via les commandes :

- docker build -t mon_application_food:tag .
- docker login
- docker tag mon_application_food:tag trevor2504/mon_application_food:tag
- docker push trevor2504/mon_application_food:tag

5. Schéma d’exécution

Pour exécuter cette application, il faut d’abord télécharger son image via docker avec la commande :

docker pull trevor2504/mon_application_food:tag

Et par la suite l’exécution se fait avec la commande :

docker run -d -p 80:80 trevor2504/mon_application_food:tag

7. Algorithme allergie

Voici un algorithme simple pour déterminer la probabilité qu'une nourriture provoque une allergie à une personne, en fonction du nombre de fois qu'elle a mangé cette nourriture et du nombre de fois où cela a entraîné une allergie.

Algorithme

1. **Entrées :**
    - n: nombre total de fois que la personne a mangé la nourriture.
    - a: nombre de fois que la nourriture a provoqué une allergie.
2. **Calcul de la probabilité :**
    - Si n est égal à 0, alors la probabilité est indéfinie (ou 0% par défaut).
    - Sinon, la probabilité P que cette nourriture provoque une allergie est calculée comme suit :
      P=na
3. **Sortie :**
    - Retourner la probabilité P.

Pseudocode

plaintext

```
fonction calculerProbabilite(n, a):
    si n == 0 alors:
        retourner "Indéfini"  // ou retourner 0
    fin si

    P = a / n
    retourner P
fin fonction
```

Exemple d'Utilisation

- Si une personne a mangé une nourriture 10 fois (n = 10) et a eu une allergie 2 fois (a = 2), la probabilité serait :
  P = \frac{2}{10} = 0.2 \text{ (ou 20%)}

Cet algorithme simple vous permet d'évaluer rapidement la probabilité d'une allergie en fonction des données fournies.

8. Conclusion

Cette application permet une gestion simple et efficace des données relatives aux personnes, nourritures et ingrédients via une API Flask. L'utilisation de Docker facilite le déploiement et la gestion de l'environnement de développement. Le diagramme de classe et le schéma d'exécution clarifient la structure de l'application et son fonctionnement.Elle permettra ainsi aux utilisateurs de pouvoir mieux contrôler leur alimentation.

Lien du github pour le code source :

https://github.com/TREVOR-TWYT/mon_application_food.git