

**Projet SIN I2 - 2023**

**Projet d’API cuisine**

|  |  |
| --- | --- |
| **ETUDIANTS:** | SHILI Ibrahim  JETUR DELABY Valentin |
| **ENSEIGNANTS:** | JANURA  HERLANT |

|  |  |  |
| --- | --- | --- |
|  | **ETABLISSEMENT HABILITE PAR LA COMMISSION DES TITRES D’INGENIEUR ET MEMBRE DE LA CONFERENCE DES GRANDES ECOLES** |  |

**14 quai de la Somme – BP 10100**  
**80082 AMIENS CEDEX 2**  
**Tél. : 03.22.66.20.00 – Fax : 03.22.66.20.10**  
**https://amiens.unilasalle.fr/**

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

Plan :

1. Introduction

* Présentation du projet et de son objectif

2. Description de l'API

* Définitions de l'API
* Présentation des différentes routes/End points de l'API

3. Outils utilisées

* Outils du projet

4. Conception et architecture de l'API

* Modèles de données
* Routes/End-points

5. Implémentation

* Détails sur l'implémentation de chaque fonctionnalité demandée
* Utilisation du Framework Flask et SQLAlchemy

6. Contraintes et problèmes rencontrés

* Implémenter les routes pour avoir les bonnes fonctionnalités
* Beaucoup de temps à comprendre les nouvelles notions (API, FLASK, environnement.)
* Autoapprentissage avantage et désavantage

7. Conclusion

* Bilan global du projet

8. Annexes

* Liens vers le dépôt GitHub et le site web (le cas échéant)

1.Introduction :

Ce rapport détaille la création d'une application pour découvrir et partager des recettes de cuisine. Notre objectif était de construire une interface permettant de créer des utilisateurs, des ingrédients et des recettes, puis de les rassembler en menus complets.

L'outil que nous avons développé est une API, une sorte de pont invisible qui permet aux utilisateurs de voir les recettes disponibles, de choisir des ingrédients en fonction du nombre de convives et de sauvegarder leurs recettes favorites. Nous avons utilisé des outils spécifiques, comme Flask et SQLAlchemy, pour construire cette API. Ce rapport explique les différentes étapes que nous avons suivies, les choses qui ont marché et celles qui ont été plus compliquées, et fournit des instructions pour utiliser notre création.

En somme, ce projet était pour nous une découverte de la partie back end du développement informatique et de la notion d’API.

2. Description de l’API :

* + - **Définition d’un API :**

Ce projet nous a permis d’acquérir la notion d’un API, qui nous a été très utile pour compléter le projet. Une API, ou Interface de Programmation Applicative, est un ensemble de règles et de protocoles permettant à différents logiciels ou applications de se parler. Elle fonctionne comme un guide qui précise comment différentes parties d'un logiciel peuvent interagir entre elles. Cela permet à un programme de demander des informations ou d'exécuter des actions auprès d'un autre programme de manière organisée et conforme à des normes définies. En somme, un API facilite la communication et l'échange d'informations entre différentes parties d'un système informatique.

* + - **Définition d’un end-point / routes :**

Un end-point, également appelé route, est une URL spécifique (Uniform Resource Locator) ou un point d'accès spécifique à une API. Chaque end-point correspond à une fonctionnalité ou à une action particulière que l'API peut réaliser.

Chaque end-point est associé à une méthode HTTP spécifique (comme GET, POST, PUT, DELETE) et définit le comportement attendu lorsque des requêtes sont envoyées à cette adresse. Les requêtes à ces end-points permettent d'accéder, de créer, de mettre à jour ou de supprimer des données spécifiques en fonction de l'action définie pour cet end-point. Dans le contexte de notre projet Postman va pouvoir nous fournir une plateforme où on pourra faire des requêtes HTTP. On n’aura pas besoin d’un front-end.

* + - **Fonctionnalités de l'API :**

L'API que nous avons développée offre un ensemble complet de fonctionnalités pour interagir avec les recettes, les ingrédients et les utilisateurs. Voici une explication détaillée des principales fonctionnalités offertes :

**1. Gestion des utilisateurs :** L'API permet la création de nouveaux utilisateurs avec des fonctionnalités telles que l'enregistrement, la connexion et la gestion du profil. L’URL associé est ‘/users/’.

**2. Gestion des ingrédients :** Les utilisateurs peuvent créer de nouveaux ingrédients, les modifier, les supprimer et les consulter. L’end point associés à cette fonctionnalité est '/ingredients/'.

**3. Création de recettes :** Les recettes peuvent être créées en associant des ingrédients existants, décrivant les étapes de préparation et spécifiant le type de plat (entrée, plat principal, dessert). L’end-point pour cette fonctionnalité est `/recettes`.

**4. Construction de menus** **:** Les menus peuvent être assemblés en regroupant des recettes préalablement enregistrées. L’end-point associés à cette fonctionnalité est `/menus/`.

**5. Consultation des ingrédients :** Les utilisateurs peuvent obtenir une liste de tous les ingrédients disponibles via l'end point `/ingredients`.

**6. Exploration des recettes :** Les recettes peuvent être consultées via des end points comme `/recettes `, permettant aux utilisateurs de parcourir toutes les recettes disponibles.

**7. Gestion des recettes préférées :** Les utilisateurs peuvent marquer des recettes comme favorites pour un accès rapide. L’URL associé est '/users/<string:user\_name>/recette\_favorite/'.

3. Outils utilisés :

**Framework Flask :** Flask est un Framework web léger et flexible pour Python. il permet de construire des applications web de manière simple et rapide, idéal pour le développement de notre API.

**SQLAlchemy :** SQLAlchemy est une bibliothèque pour Python qui facilite l'interaction avec les bases de données relationnelles. Nous l'avons utilisé pour gérer et manipuler les données de notre application, offrant ainsi une intégration facile avec Flask.

**Postman :** Postman est une application qui permet de tester et de travailler avec des API. Nous l'avons utilisé pour tester les différentes fonctionnalités de notre API de manière efficace.

**PyCharm :** PyCharm est un environnement de développement intégré (IDE) conçu spécifiquement pour les développeurs utilisant le langage de programmation Python. Pourquoi a-t-on choisi PyCharm ? Parce qu’en réalité PyCharm simplifie la création et la gestion des environnements virtuels en nous offrant une interface conviviale pour configurer et gérer les interpréteurs Python spécifiques à notre projet. De plus, en utilisant l’outil virtualenv il nous permets de gérer la création et l'activation de ces environnements isolés.

4. Conception et architecture de l'API

Dans le cadre de la conception de notre API, ce code initial configure l'environnement de base nécessaire à l'utilisation de Flask, un Framework web, ainsi que de SQLAlchemy, un outil facilitant l'interaction avec une base de données SQLite.

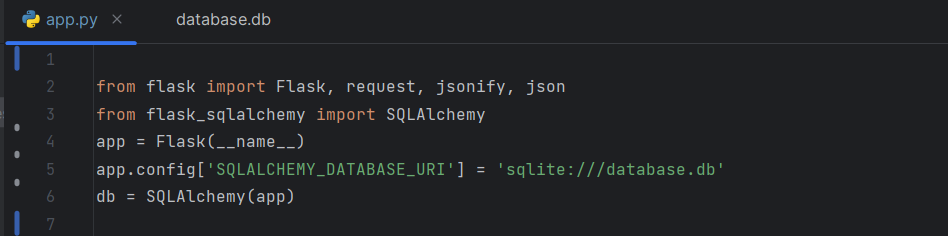
Voici une explication plus détaillée de chaque étape :

**Importation des modules :** La première section du code concerne l'importation des modules essentiels. Flask est utilisé pour créer l'infrastructure de l'API web, tandis que SQLAlchemy facilite la communication avec la base de données. Les modules supplémentaires comme request, jsonify, et json sont importés pour gérer les requêtes et les données au format JSON.

**Création de l'application Flask :** L'instruction ‘app = Flask(\_\_name\_\_)’ instancie une application Flask, constituée de routes, de fonctionnalités et de logiques spécifiques à notre API. C'est dans cette application que seront définies les différentes fonctionnalités de notre service, telles que la création d'utilisateurs, la gestion des recettes, etc.

**Configuration de la base de données :** La ligne app.config['SQLALCHEMY\_DATABASE\_URI'] configure l'adresse et le type de la base de données à utiliser. Dans notre cas, SQLite est choisi avec le fichier ‘database.db’ pour stocker nos données. Cette configuration détermine comment l'application se connecte à la base de données.

**Initialisation de SQLAlchemy :** En utilisant ‘db = SQLAlchemy(app)’, une instance de SQLAlchemy est créée et liée à notre application Flask. Cette liaison permet à SQLAlchemy d'interagir avec la base de données en utilisant les paramètres spécifiés dans la configuration de l'application.



Cette portion du code établit les fondations essentielles pour notre application API. Elle prépare l'environnement nécessaire pour définir les modèles de données(classes), les end-points (routes) et les opérations de gestion des données. Cette configuration initiale nous permettra de construire et de déployer des fonctionnalités plus avancées pour notre API

* + - **Modèles de données :**

On a défini dans notre programme plusieurs structures de données afin de représenter les utilisateurs, les recettes, les ingrédients, les menus. En effet les classes dans notre programme s’expliquent comme suit :

‘User’ : Représente un utilisateur avec un identifiant unique (‘id’) et un nom unique (‘name’). Chaque utilisateur peut être associé à ses recettes favorites.

‘Ingredient’ : Modélise un ingrédient avec un identifiant unique (‘id’) et un nom unique (‘name’). Il est associé à des recettes via une relation many-to-many.

‘Recette’ : Représente une recette avec un nom (‘name’), un type (‘type’), et une liste d'ingrédients (‘ingredients’). La relation many-to-many avec les ingrédients est établie via la table recette\_ingredient.

Menu : Modélise un menu avec un nom (‘name’) et des plats tels qu'une entrée, un plat principal et un dessert. Les plats sont référencés par des clés étrangères (‘entry\_id’, ‘main\_dish\_id’, ‘dessert\_id’) associées à la table Recette. recette\_ingredient : Représente la table de liaison entre les recettes et les ingrédients, incluant la quantité (‘quantity’) d'ingrédients dans chaque recette.

‘Step’ : Modélise les étapes d'une recette avec une description (description). Les étapes sont liées aux recettes via un champ de texte (‘recette\_name’) correspondant au nom de la recette.

RecetteFavorite : Représente les recettes favorites d'un utilisateur. Chaque entrée dans cette table contient l'id de la recette et l'id de l'utilisateur.

**La relation many-to-many :**

La relation many-to-many (plusieurs-à-plusieurs) est une relation entre deux entités où plusieurs éléments d'une entité peuvent être liés à plusieurs éléments d'une autre entité, et vice versa. Dans le contexte du projet, la table de liaison recette\_ingredient permet de gérer many-to-many entre les recettes et les ingrédients. Cette table agit comme un lien intermédiaire pour associer les recettes aux ingrédients et enregistrer des informations supplémentaires, comme la quantité d'un ingrédient utilisée dans une recette spécifique.

* + - **Routes/End-points et logique de contrôle :**

Les end-points définis dans l'API permettent d'interagir avec différentes fonctionnalités telles que la création, la récupération et la suppression des utilisateurs, des ingrédients, des recettes, des étapes de recettes, des recettes favorites et des menus. Chaque end-point a été conçu pour offrir une interaction claire avec l'API.

Création d'un Utilisateur

L'end-point `'/users/'` permet la création d'un nouvel utilisateur via une requête POST. L'utilisateur est créé en vérifiant la présence du nom d'utilisateur fourni dans le formulaire de la requête. Si les données sont valides, un nouvel utilisateur est ajouté à la base de données. La logique de contrôle vérifie si le nom d'utilisateur est fourni dans les données du formulaire (username = request.form.get('username')). Si le nom est présent, un nouvel utilisateur est créé avec ce nom et ajouté à la base de données.

Récupération de la Liste des Utilisateurs

Le même end-point `'/users/'` avec la méthode GET permet de récupérer la liste de tous les utilisateurs existants dans la base de données sous forme de réponse JSON\*. La requête GET est utilisée pour récupérer tous les utilisateurs de la base de données (users = User.query.all()). Ensuite, une liste de dictionnaires contenant l'ID et le nom de chaque utilisateur est retournée sous forme de réponse JSON.

Suppression d'un Utilisateur

Pour supprimer un utilisateur, l'end-point `'/users/'` est utilisé avec la méthode DELETE. L'utilisateur à supprimer est identifié par son nom dans le formulaire de la requête. Pour supprimer un utilisateur, la logique vérifie si le nom d'utilisateur est fourni dans les données de la requête DELETE. Si oui, elle recherche l'utilisateur par son nom dans la base de données et le supprime s'il est trouvé.

Création d'un Ingrédient

L'end-point `'/ingredients/'` permet la création d'un nouvel ingrédient via une requête POST. Le nom de l'ingrédient est extrait des données du formulaire de la requête. Si toutes les données requises sont présentes, un nouvel ingrédient est ajouté à la base de données. La logique de contrôle extrait le nom de l'ingrédient des données du formulaire. Si le nom est présent, un nouvel ingrédient est créé avec ce nom et ajouté à la base de données.

Récupération de la Liste des Ingrédients

L'end-point `'/ingredients/'` avec la méthode GET permet de récupérer la liste de tous les ingrédients existants dans la base de données sous forme de réponse JSON. Cette opération est similaire à la récupération des utilisateurs, récupérant tous les ingrédients de la base de données et renvoyant une liste d'objets contenant l'ID et le nom de chaque ingrédient.

Suppression d'un Ingrédient

Pour supprimer un ingrédient, l'end-point `'/ingredients/'` est utilisé avec la méthode DELETE. L'ingrédient à supprimer est identifié par son nom dans le formulaire de la requête. Pour supprimer un ingrédient, la logique vérifie si le nom de l'ingrédient est fourni dans les données de la requête DELETE. Si trouvé, l'ingrédient est supprimé de la base de données. La logique vérifie si toutes les données nécessaires pour créer une recette sont présentes dans la requête POST. Elle convertit les données JSON des ingrédients en une liste de dictionnaires, crée une nouvelle recette et associe les ingrédients à cette recette en les ajoutant à la base de données.

Création d'une Recette

L'end-point `'/recettes/'` permet la création d'une nouvelle recette via une requête POST. Les informations de la recette (nom, type, ingrédients) sont fournies dans le formulaire de la requête. La logique vérifie si toutes les données nécessaires pour créer une recette sont présentes dans la requête POST. Elle convertit les données JSON des ingrédients en une liste de dictionnaires, crée une nouvelle recette et associe les ingrédients à cette recette en les ajoutant à la base de données.

Récupération de la Liste des Recettes

L'end-point `'/recettes/'` avec la méthode GET permet de récupérer la liste de toutes les recettes existantes dans la base de données sous forme de réponse JSON. Similaire à la récupération des utilisateurs et des ingrédients, elle récupère toutes les recettes de la base de données et renvoie une liste d'objets contenant des détails sur chaque recette.

JSON : acronyme de "JavaScript Object Notation", est un format de données léger, lisible par les humains et facile à comprendre pour les machines. Il est largement utilisé pour échanger des données entre un serveur et un client web.

5. Implémentation

Dans le cadre de l'implémentation des fonctionnalités demandées, on a été utilisés le Framework Flask et SQLAlchemy conjointement pour mettre en place les différentes opérations de l'API.

Utilisation du Framework Flask :

- Gestion des Requêtes : Flask permet de gérer différents types de requêtes HTTP (GET, POST, DELETE) vers ces endpoints en utilisant les décorateurs (`@app.route()`) pour définir les actions à effectuer en fonction du type de requête reçu.

- Extraction des Données : Les données nécessaires sont extraites des requêtes (par exemple, le nom d'utilisateur, le nom d'ingrédient, les détails de la recette à l'aide de `request.form.get()` ou `request.json.get()` selon le format des données(données form ou JSON).

Utilisation de SQLAlchemy :

- Modèles de Données : SQLAlchemy est utilisé pour définir les modèles de données (`User`, `Ingredient`, `Recette`, `Menu`, etc.) qui représentent les tables de la base de données. Chaque classe modèle correspond à une table dans la base de données et est configurée avec des colonnes et des relations entre les tables (relations many-to-many).

- Manipulation de la Base de Données : Les opérations de création, lecture, mise à jour et suppression sur la base de données sont gérées via les méthodes de SQLAlchemy (`db.session.add()`, `db.session.delete()`, `db.session.commit()`).

- Relations entre les Tables : Les relations entre les modèles (comme les recettes ayant plusieurs ingrédients, les utilisateurs ayant des recettes préférées, etc.) sont définies à l'aide des attributs `db.relationship()’ et des tables de liaison `db.Table()` pour les relations many-to-many.

Implémentation des Fonctionnalités :

Chaque fonction est conçue pour valider les données, interagir avec la base de données pour exécuter les actions demandées et renvoyer des réponses appropriées (codes de statut HTTP et messages JSON) en fonction du résultat de l'opération.

L'association de Flask et SQLAlchemy facilite la création, la gestion et la manipulation des données pour les différentes fonctionnalités de l'API tout en garantissant la cohérence des données stockées dans la base de données.

6. Contraintes et problèmes rencontrés

Malgré les réussites qu’on a réalisées, on a eu plusieurs défis et difficultés. La première difficulté qui s’est présenté c’était de bien comprendre les nouveaux concepts. Tels que l’API, SQLAlchemy, les routes et l’environnement virtuel. On a passé beaucoup trop de temps à chercher les bonnes sources d’informations et à comprendre ces notions.

L’autoapprentissage était pour nous un défi à relever, on n’était pas guidée du tout, on devait tout faire par nous-mêmes. Parfois c’était très décourageant, mais on pense qu’à la fin ça nous a bien aider à nous développer sur le plan personnel. On a acquéri au final des compétences techniques qui nous servira non seulement dans notre métier d’ingénieur mais aussi dans notre vie.

Sur le plan pratique on a eu du mal à conditionné les routes afin d’assurer les différentes requêtes de l’utilisateur. Notamment pour lister les recettes favorites et les associer à un utilisateur unique.

7. Conclusion :

Ce projet a représenté une exploration enrichissante dans le développement d'une API utilisant Flask et SQLAlchemy pour la gestion de recettes.

On a réussi à implémenter les fonctionnalités principales, permettant la création, la récupération, la mise à jour et la suppression des données relatives aux utilisateurs, aux recettes, aux ingrédients, aux menus et aux étapes de recette. On a bien défini les endpoints pour offrir une interaction claire et structurée avec la base de données.

Malgré, les défis qu’on a rencontrés tout au long du processus, ce projet à bien rempli son rôle. Il nous a plongé dans le monde de développement back-end et nous a donner l’opportunité de créer notre propre API, ce qui a été très enrichissant.