Présentation de l’application du restaurant (exercice de recrutement)

Architecture de l’application :

L’application est construite en C# (dot net core 7). J’ai mis en place un fonctionnement sous forme de service windows permettant de facilement gérer l'état de l’application dans un environnement windows. Un serveur web autonome Kestrel est automatiquement lancé par le service permettant de recevoir des appels HTTP REST API (appels qui seraient idéalement réalisés depuis un futur serveur pour le front end). Pour le test, on peut utiliser postman.

Pour la base de données, il m’a semblé cohérent d’utiliser l’ORM Entity Framework et une base de données SqLite. Dans un réel environnement de production, on devrait plutôt utiliser un serveur SQL dédié, mais dans notre cas cela est plus logique.

La base de données SQL est composée d'une table restaurant, une table plat et une table assurant la liaison entre les restaurants et les plats.

Restaurants :

	Id	Name
1	56bf802e-afcf-4110-9aa2-28562f35296d	restaurant1
2	b272b1aa-5a69-41e2-a3d1-80040f87be3c	restaurant2
3	286a92ed-1fbb-4562-a473-ddcb1a62559c	restaurant3

Meals :

	Id	Name	Image
1	52771	Spicy Arrabiata Penne	https://www.themealdb.com/images/media/meals/ustsqw1468250014.jpg
2	53014	Pizza Express Margherita	https://www.themealdb.com/images/media/meals/x0lk931587671540.jpg
3	53010	Lamb Tzatziki Burgers	https://www.themealdb.com/images/media/meals/k420tj1585565244.jpg

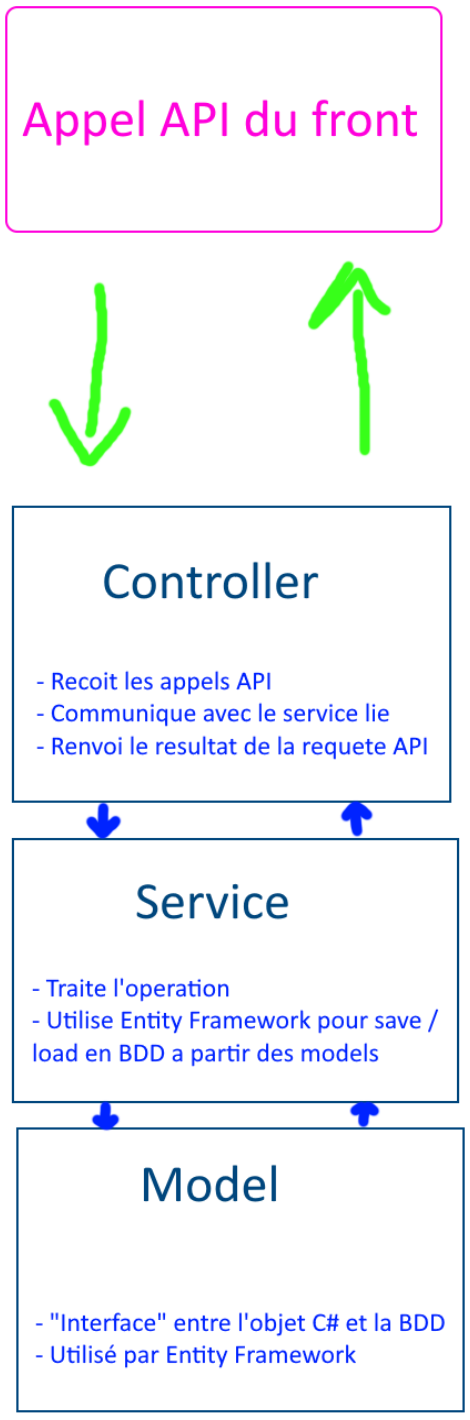
Liaison :

	MealsId	RestaurantsId
1	52771	56bf802e-afcf-4110-9aa2-28562f35296d
2	53014	56bf802e-afcf-4110-9aa2-28562f35296d
3	53014	b272b1aa-5a69-41e2-a3d1-80040f87be3c
4	53010	b272b1aa-5a69-41e2-a3d1-80040f87be3c
5	53010	286a92ed-1fbb-4562-a473-ddcb1a62559c

Voici comment j'ai structuré le code de l'application avec tous les dossiers et fichiers expliqués :
(Legende - Dossier, Classe)

- Program : Le lancement de l'application, crée le service et le serveur web.
- Worker : Classe gérant la durée de vie du service et permettant de le stopper par un signal.
- LogManager : Classe gérant l'affichage des logs dans un fichier + console
- UseCustomError : Classe permettant d'afficher des erreurs personnalisées au client pour les appels API
- Controllers : Ici se situent tous les appels API REST web
    - RestaurantController : Les appels API liés aux restaurants
    - MealController : Les appels API directement liés aux plats
- Models : Les modèles des objets correspondant à la base de donnée
    - Restaurant : Le modèle lié à la table Restaurants
    - Meal : Le modèle lié à la table Meals
    - RestaurantsContext : La classe de contexte nécessaire pour Entity Framework afin de définir les liaisons entre les objets
- Services : Les services effectuent les opérations principales, et font le lien avec la bdd via entity framework
    - MealApiService : effectue les appels à l'api the meal db
    - MealService : service lié au modèle des plats
    - RestaurantService : service lié au modèle des restaurants
- Database : Ce dossier contient la base de données locale Sqlite.

Voici le lien entre chaque couche du code :



Installation et utilisation :

Pour installer l'application, il faut charger le fichier "ExerciceRecrutement/ExerciceRecrutement.sln" dans visual studio. Ensuite, on peut compiler l'application et la lancer directement. Une console doit normalement s'ouvrir et lancer le service principal. Il devrait s'afficher "started service" dans la console si tout est OK.

Pour tester les différents appels API, on peut utiliser le client Postman. Voici les différents appels possibles :

Afficher la liste des restaurants : (Fonctionnalité 1)
GET http://localhost:8080/api/Restaurant

Afficher les plats d'un restaurant : (Fonctionnalité 3)
GET http://localhost:8080/api/Restaurant/[RESTAURANT_ID]

Ajouter un restaurant : (Fonctionnalité 2)
POST http://localhost:8080/api/Restaurant
body :
```
{
   "name": "[NEW_RESTAURANT_NAME]"
}
```

Chercher un nouveau plat par son nom : (Fonctionnalité Bonus)
GET http://localhost:8080/api/Meal/[MEAL_NAME]

Ajouter un nouveau plat a un restaurant : (Fonctionnalité 5)
POST http://localhost:8080/api/Restaurant/[RESTAURANT_ID]
body :
```
{
   "meal": "[MEAL_NAME]"
}
```

Spécification des futurs développements :

- Ajouter de nouveaux restaurants - Fonctionnalité 2 :

J’ai déjà implémenté cette fonctionnalité afin de rendre l’application vraiment utilisable. Cependant, si je devais donner des spécifications à un autre développeur, voici ce que je formulerai :

- Créer un nouvel appel API dans le controller du restaurant POST avec un paramètre “nom” dans le body.
- Créer une nouvelle méthode pour créer un restaurant dans le service restaurant. Utiliser Entity Framework pour ajouter un nouvel objet restaurant avec le nom spécifié à la BDD.
- Retourner un message de succès ou d’erreur au client selon le résultat.

- Afficher une vue détaillée d’un plat (Nom, Image, Catégorie, Ingrédients et quantités) - Fonctionnalité 4 :

Je n’ai pas implémenté cette fonctionnalité, mais la réalisation de celle-ci pourrait être ainsi :

- Ajouter des propriétés Catégorie (Text), Ingrédients (ManyToMany) et quantités (Number) au modèle “Meal”
- Creer un Modele Catégories et Ingrédients
- Effectuer la liaison en BDD des différents modèles
- Modifier la méthode d’ajout de plat pour prendre en compte ces différentes propriétés
- Créer un nouvel appel API dans le controller des plats GET avec un paramètre id des plats
- Créer une nouvelle méthode pour renvoyer un plat spécifique et toutes ses propriétés dans le service des plats

- Créer un serveur front end - Fonctionnalité Bonus :

Dans le futur, si je devais gérer le front, je pense que je choisirais de mettre en place un serveur NodeJs pour le front. Je pourrai utiliser un framework comme ViewJs pour l’affichage des vues HTML.

NodeJs aurait un serveur API comme express pour récupérer les pages web. Dans le code client de la page, des appels HTTP seraient effectués à notre API Back End. Ainsi on rend le back end complètement indépendant du front ce qui me semble beaucoup plus facile à gérer et pour simplifier le développement des deux parties.

Merci de votre lecture, et n'hésitez pas à revenir vers moi si vous avez des questions ou avez besoin de plus de précisions sur ce travail.