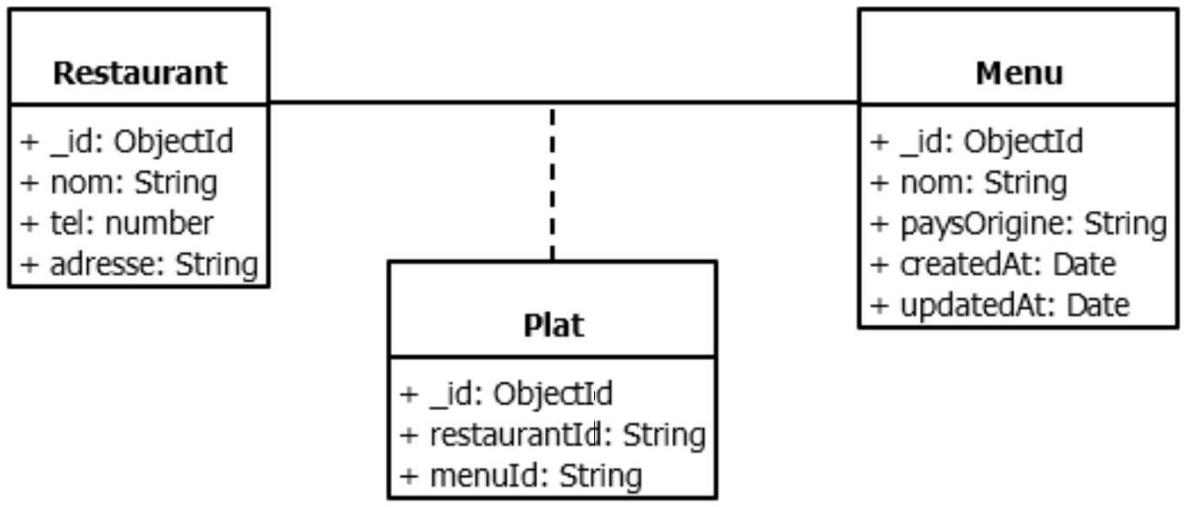
esprit Se former autrement HONORIS UNITED UNIVERSITIES	**EXAMEN** Semestre : 1 ☐ 2 ■ Session : Principale ☐ Rattrapage ■
Module : Gaming Software Engineering application côté serveur Enseignant(s) : Equipe mobile Classe(s) : 4 GamiX	
Documents autorisés : OUI ■ NON ☐ Nombre de pages : 8 Calculatrice autorisée : OUI ☐ NON ■ Internet autorisée : OUI ☐ NON ■	
Date : 21/06/2022 Heure : 12h30 Durée : 01h30	

*** La validation tient compte seulement de l'exécutable de l'application. Elle se fera via une collection Postman apportée par le validateur sur clé USB lors de la validation.**
*** Il faut respecter l'utilisation des noms (attributs, classes, routes, etc.) utilisés dans l'énoncé.**
*** Vous n'êtes pas demandé de faire un « build » ou un « run » au fichier Dockerfile et docker-compose.**
*** L'application Node.js doit être en ECMAScript (pas CommonJS, sinon -2 de la note finale) et tourner sur le port 9090.**

L'objectif est de réaliser des services web pour une application mobile qui permet à ses utilisateurs de chercher un menu dans un restaurant.

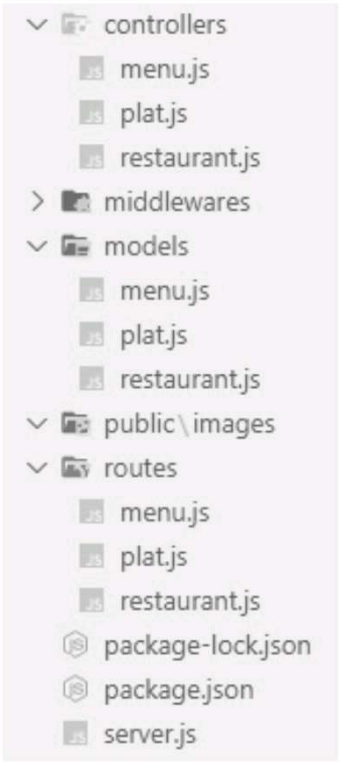
Travail demandé

Soit le diagramme de classes suivant :



Partie 1 (3 pts)

1. Initialisez votre projet Node.js.
2. Créez la structure suivante :



3. Configurez la connexion à la base de données nommée « **examen4gamix2122sr** » dans le fichier « **server.js** ».
4. Implémentez les classes du diagramme de classes comme étant des modèles « **mongoose** ».

Partie 2 (2,5 pts)

1. Utilisez les middlewares suivants dans votre application :
    - « **morgan** » avec l'option « **dev** ».
    - « **cors** ».
    - Parsing JSON du corps de la requête.
2. Créez un middleware personnalisé qui permet de retourner le **code 404** et un JSON avec la clé « **message** » et comme valeur « **Not Found** » lorsque l'utilisateur fait appel à une route qui n'existe pas ou s'il n'utilise pas la méthode HTTP correcte sur une route existante.

Partie 3 (9,5 pts)

Implémentez les routes suivantes :

Description :	Permet d'ajouter un nouveau restaurant		
Méthode HTTP :	POST	**URL :**	/restaurants
Contraintes :	- La longueur de la chaine de caractères des attributs « nom » et « adresse » doit être comprise entre 3 et 30. - L'attribut « tel » doit être composé de 8 chiffres.		

Exemple de corps la de requête :	{
	"nom": "Ha'Food",
	"tel": 99999999,
	"adresse": "Ariana Soghra"
	}		

Exemple de réponse de la requête (succès) :	{
	"nom": "Ha'Food",
	"tel": 99999999,
	"adresse": "Ariana Soghra"
	}		

Code de retour en cas de succès :	201	**Code(s) de retour en cas de non-succès :**	- 400 : si au moins l'une des contraintes n'est pas respecté. - 500 : problème au niveau du serveur.

Description :	Permet de récupérer la liste des restaurants		
Méthode HTTP :	GET	**URL :**	/restaurants
Contraintes :	Aucune contrainte.		
Exemple de corps la de requête :	Aucun corps.		

Exemple de réponse de la requête (succès) :	[
	{
	"_id": "62af1aa924da77b17e2adc8d",
	"nom": "Ha'Food"
	}
	]		

Code de retour en cas de succès :	200	**Code(s) de retour en cas de non-succès :**	- 500 : problème au niveau du serveur.

Description :	Permet de récupérer un restaurant à l'aide de son id		
Méthode HTTP :	GET	**URL :**	/restaurants/:id
Contraintes :	Aucune contrainte.		
Exemple de corps la de requête :	Aucun corps.		

Exemple de réponse de la requête (succès) :	{
	"_id": "62af1aa924da77b17e2adc8d",
	"nom": "Ha'Food",
	"tel": 99999999,
	"adresse": "Ariana Soghra",
	"__v": 0
	}		

Code de retour en cas de succès :	200	**Code(s) de retour en cas de non-succès :**	- 500 : problème au niveau du serveur.

Description :	Permet d'ajouter un nouveau menu		
Méthode HTTP :	POST	**URL :**	/menus
Contraintes :	- La longueur de la chaine de caractères des attributs « nom » et « paysOrigine » doit être comprise entre 3 et 30.		

Exemple de corps la de requête :	{
	"nom": "Kaftaji",
	"paysOrigine": "Tunisie"
	}		

Exemple de réponse de la requête (succès) :	{
	"nom": "Kaftaji",
	"paysOrigine": "Tunisie"
	}		

Code de retour en cas de succès :	201	**Code(s) de retour en cas de non-succès :**	- 400 : si au moins l'une des contraintes n'est pas respecté. - 500 : problème au niveau du serveur.

Description :	Permet de récupérer la liste des menus		
Méthode HTTP :	GET	**URL :**	/menus
Contraintes :	Aucune contrainte.		
Exemple de corps la de requête :	Aucun corps.		

Exemple de réponse de la requête (succès) :	[
	{
	"_id": "62af201124da77b17e2adc91",
	"nom": "Kaftaji"
	}
	]		

Code de retour en cas de succès :	200	**Code(s) de retour en cas de non-succès :**	- 500 : problème au niveau du serveur.

Description :	Permet de récupérer un menu à l'aide de son id		
Méthode HTTP :	GET	**URL :**	/menus/:id
Contraintes :	Aucune contrainte.		
Exemple de corps la de requête :	Aucun corps.		

Exemple de réponse de la requête (succès) :	{
	"_id": "62af201124da77b17e2adc91",
	"nom": "Kaftaji",
	"paysOrigine": "Tunisie",
	"createdAt": "2022-06-19T13:09:37.265Z",
	"updatedAt": "2022-06-19T13:09:37.265Z",
	"__v": 0
	}		

Code de retour en cas de succès :	200	**Code(s) de retour en cas de non-succès :**	- 500 : problème au niveau du serveur.

Description :	Permet d'ajouter un nouveau menu à un restaurant		
Méthode HTTP :	POST	**URL :**	/plats/:restaurant/:menu
Contraintes :	Aucune contrainte.		
Exemple de corps la de requête :	Aucun corps.		

Exemple de réponse de la requête (succès) :	{
	"restaurantId": "62af1aa924da77b17e2adc8d",
	"menuId": "62af201124da77b17e2adc91"
	}		

Code de retour en cas de succès :	201	**Code(s) de retour en cas de non-succès :**	- 500 : problème au niveau du serveur.

Description :	Permet de récupérer la liste des menus d'un restaurant donné		
Méthode HTTP :	GET	**URL :**	/plats/:restaurant
Contraintes :	Aucune contrainte.		
Exemple de corps la de requête :	Aucun corps.		

Exemple de réponse de la requête (succès) :	[
	{
	"_id": "62af23a424da77b17e2adc95",
	"restaurantId": "62af1aa924da77b17e2adc8d",
	"menuId": "62af201124da77b17e2adc91",
	"__v": 0
	}
	]		

Code de retour en cas de succès :	200	**Code(s) de retour en cas de non-succès :**	- 500 : problème au niveau du serveur.

Description :	Permet de supprimer un plat d'un restaurant		
Méthode HTTP :	DELETE	**URL :**	/plats/:restaurant/:menu
Contraintes :	Aucune contrainte.		
Exemple de corps la de requête :	Aucun corps.		

Exemple de réponse de la requête (succès) :	{
	"_id": "62af23a424da77b17e2adc95",
	"restaurantId": "62af1aa924da77b17e2adc8d",
	"menuId": "62af201124da77b17e2adc91",
	"__v": 0
	}		

Code de retour en cas de succès :	200	**Code(s) de retour en cas de non-succès :**	- 500 : problème au niveau du serveur.

Partie 4 (5 pts)

Afin de déployer notre application sur un serveur, nous avons besoin de conteneuriser la solution. Pour ceci, on se propose de créer deux services interconnectés, le premier conteneur est celui la base de données MongoDB et le deuxième celui de l'application Node.js.

Pour ce faire, vous allez créer un Dockerfile ainsi qu'un docker-compose avec les spécifications suivantes :

Pour le Dockerfile (1,5 pts)

- Le mainteneur est GSE GAMIX avec l'adresse email : « gse.gamix @esprit.tn ».

- L'utilisateur du conteneur doit être « node » et non pas « root ».

- Le mode de déploiement de Node.js est le mode production.

- Le code source du projet dans être mis sous le dossier « /home/node/app ».

- Exposer le port 3000.

Pour le docker-compose (3,5 pts)

Contient deux services :

1. **rest :**
    - Le nom de l'image est « examen-gse-gamix-sr-2122 ».
    - Le tag de l'image est « 1 ».
    - La variable d'environnement « NODE_ENV » est configurer à « production ».
    - Le port de ce service est « 3030 ».
    - Connecté à l'interface réseau appelé « gamix2122sr ».
    - Dépond du service « data ».

2. **data :**
    - Un conteneur à partir de l'image « mongo » avec le tag « 5 ».
    - Le port de ce service est « 3031 ».
    - Connecté à l'interface réseau appelé « gamix2122sr ».
    - Connecté à un volume appelé « db ».

Bonne chance