Objectif

Savoir réaliser et mettre en place une architecture REST
Cette SAÉ ne met en oeuvre que les compétences REST et ne s'occupe pas des aspects MVC

Travail à réaliser

L'entreprise `Pizzaland` souhaite mettre à disposition des entreprises tierces, la gestion de ses pizzas et leurs ingrédients ainsi que les commandes qui sont effectuées par les clients. Elle souhaite mettre en place une API REST pour assurer cette tâche.

Pour simplifier le travail, `Pizzaland` a décidé de ne communiquer qu'en JSON. Bien évidemment ce travail necessite une base de données. Pour faciliter la maintenance future de l'application on souhaite isoler les requêtes SQL dans des DAO, et la connexion à la base de données à un seul endoit.

Ce travail peut être réalisé en binôme. Tous les endpoints doivent figurer dans une collection `RESTED` Une réunion de rendu de projet sera organisée pour discuter avec chaque binôme des étapes réalisées.

Partie 1 : Les ingrédients

On s'interesse tout d'abord aux ingrédients. Les ingrédients sont caractérisés par leur id, leur nom et leur prix. Il faut bien sûr une table du SGBD pour les stocker, un DTO pour le mapping objet/relationnel et un DAOIngrédient pour assurer les requêtes.

Au minimum, les endpoints suivants doivent être assurés

`GET`	`/ingredients`	pour obtenir la collection de tous les ingrédients
`GET`	`/ingredients/{id}`	pour obtenir un ingrédient particulier
`GET`	`/ingredients/{id}/name`	pour obtenir uniquement le nom d'un ingrédient spécifique
`POST`	`/ingredients`	pour ajouter un nouvel ingrédient
`DELETE`	`/ingredients/{id}`	pour supprimer un ingrédient existant

Les cas d'erreurs doivent bien évidemment être traités.

Partie 2 : Les pizzas

On s'interesse maintenant aux pizzas. Bien évidemment les pizzas ont un nom, un type de pâte, un prixBase et sont caractérisées par des ingrédients. Un même ingrédient pouvant être utilisé dans plusieurs types de pizzas. le prix final d'une pizza est calculé par l'addition du prix de base et la somme des ingrédients qui la composent.

Au minimum, les endpoints suivants doivent être assurés

GET	/pizzas	pour obtenir la collection de toutes les pizzas
GET	/pizzas/{id}	une pizza en particulier
POST	/pizzas	ajout d'une nouvelle pizza avec ses ingrédients
DELETE	/pizzas/{id}	suppression d'une pizza
PATCH	/pizzas/{id}	modif d'un attribut d'une pizza (le prix augmente)
POST	/pizzas/{id}	ajout d'un ingrédient à une pizza
DELETE	/pizzas/{id}/{idIngredient}	suppression d'un ingrédient d'une pizza
GET	/pizzas/{id}/prixfinal	fournit le prix final de la pizza

GET sur une pizza doit renvoyer la pizza et tous ses ingrédients.

Partie 3 : les commandes

On s'occupe maintenant de la partie utilisateurs. Un utilisateur peut bien évidemment commander plusieurs pizzas dans sa commande. Chaque commande doit contenir l'identifiant de l'utilisateur, la date de commande, et les pizzas commandées.
On ne s'occupera pas de la gestion des utilisateurs. Ceux ci peuvent être mis "en dur" dans la base de données.

Au minimum, les endpoints suivants doivent être assurés

GET	/commandes	liste des commandes en cours
GET	/commandes/{id}	le détail d'une commande
POST	/commandes	enregistrement d'une nouvelle commande
GET	/commandes/{id}/prixfinal	fournit le prix final de la commande

GET sur une commande doit renvoyer la commande avec toutes ses pizzas, elles mêmes contenant tous leurs ingrédients

Partie 4 : restrictions d'Accès

Actuellement tout le monde peut faire toutes les actions. On souhaite maintenant restreindre l'accès à tous les endpoints de mise à jour (autres que GET) à des personnes qui se sont authentifiées et ont récupéré un token de type APIToken. Mettre en place un endpoint qui permet avec son login/mdp de récupérer un token et faire en sorte que ce token soit vérifié partout où c'est necessaire.

Au minimum, les endpoints suivants doivent être assurés

GET	/users/token	récupération du token d'authentification