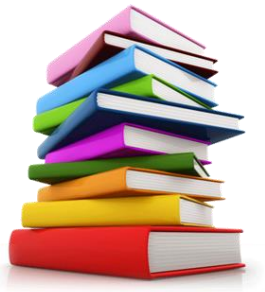
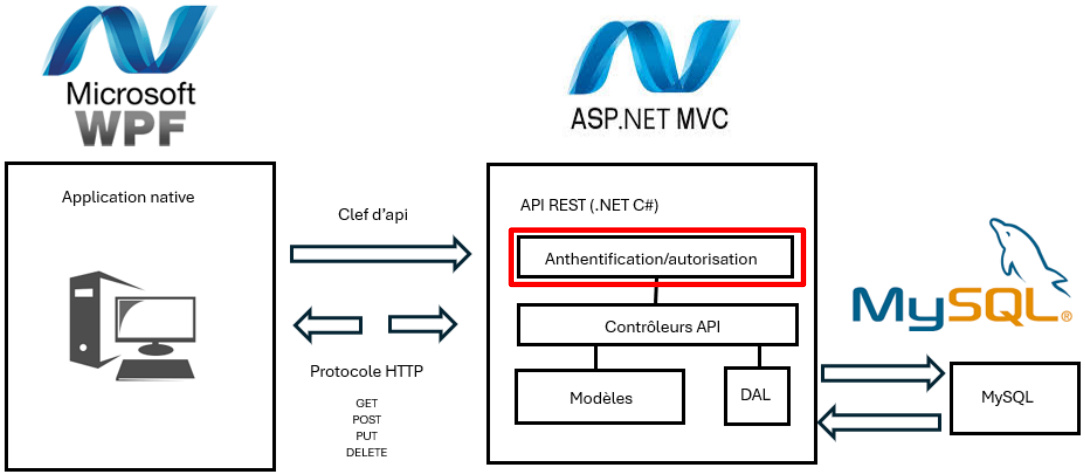
Ajout d'authentification et autorisation sur votre API REST



De façon individuelle, vous devez implémenter l'aspect sécurité de votre API REST permettant la vérification de la présence d'une clef d'API valide pour l'accès à des ressources. Comme certains services offerts par votre API sont plus « sensible », vous devez vérifier que la clef d'api fournie est liée à un rôle permettant l'accès à cette ressource.

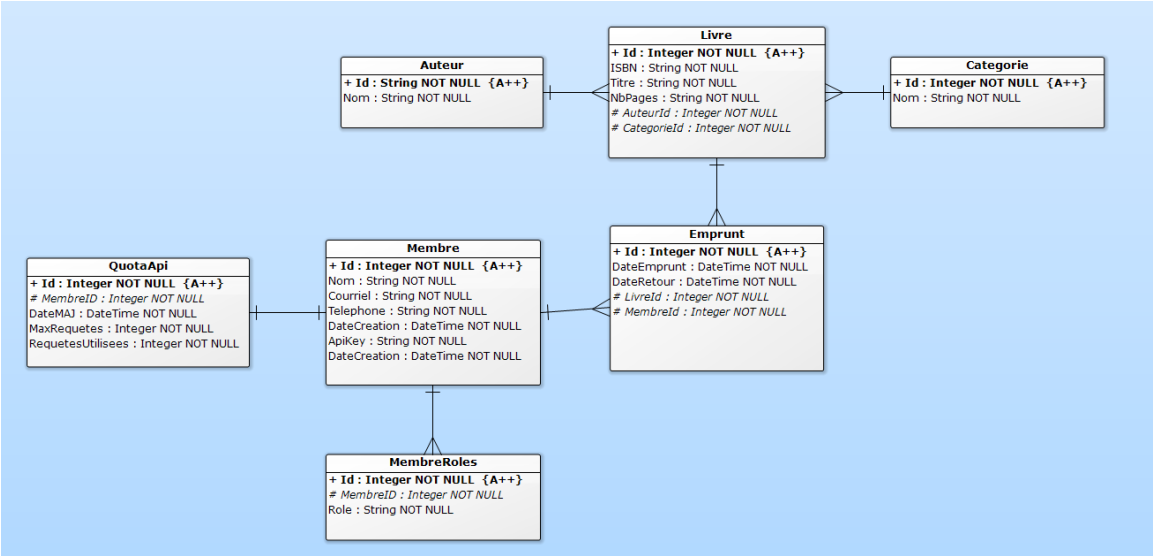
La partie 2 du projet se concentrera sur l'authentification et sur l'autorisation :



Par la même occasion, vous devez ajouter un nouvel élément à cette partie, la gestion d'un nombre de requêtes par période donnée. Pour ce faire, vous allez ajouter deux tables dans votre base de données :

1. QuotaAPI
2. MembreRoles

Voici le nouveau schéma de la base de données :



Les scripts d'initialisation ainsi que l'insertion de quelques données vous seront fournis

Ajout de contraintes d'accès à votre API

Limite d'appels par période

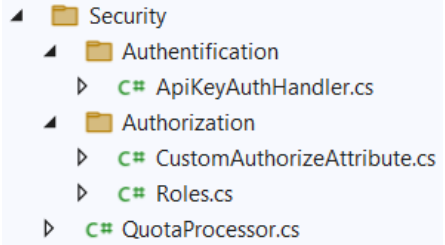
Vous devez faire l'implémentation d'une mécanique servant à limiter un utilisateur à un nombre d'appels par période. Pour rendre ceci testable, j'ai mis par défaut un maximum de 20 appels par membre. Vous devez vérifier que ces 20 appels sont faits par période de 24h. Si la période de 24h est passée, vous devez réinitialiser le champ DateMAJ à la date et l'heure actuelle, et réinitialiser le champ RequêtesUtilisées à 0. Pour centraliser cette logique, je vous invite à faire une classe l'implémentant. Afin de ne pas dupliquer cette mécanique à l'intérieur de vos routes, vous devez créer cette logique à l'intérieur de votre ApiKeyAuthHandler **ET** dans votre CustomAuthorizeAttribute.

Ajout de nouvelles classes

Vous devez ajouter ces classes à votre programme :

- Le modèle **QuotaApi**
- **QuotaApiFactory**
- **RolesFactory**

- Toute la partie sur l'authentification et l'autorisation : le dossier **Security**
- La classe servant à faire la gestion des quotas



Sécurité de votre service :

- Auteur :
  - Ajouter un nouvel auteur.
    - Nécessite d'être administrateur
  - Modifier un auteur existant.
    - Nécessite d'être administrateur ou éditeur
  - Supprimer un auteur existant.
    - Nécessite d'être administrateur
  - Récupérer les informations d'un auteur.
    - Aucun droit supplémentaire
  - Récupérer la liste complète des auteurs.
    - Aucun droit supplémentaire

- Livre :
  - Pour ce contrôleur, vous devez faire l'authentification sur l'ensemble du contrôleur.
  - Vous devez également permettre à un éditeur de modifier un livre
  - Pour l'ajout et la suppression, il faut être administrateur.
  - Vous devez ajouter l'attribut [AllowAnonymous] sur l'ensemble de routes en GET.
- Membre :
  - Ajouter un nouveau membre
    - Nécessite d'être administrateur
- Emprunt :

  Il faut être administrateur pour utiliser des ressources de ce contrôleur

Ajout de fonctionnalité

Vous devez modifier la création d'un membre pour créer en même temps une entrée dans la table projet_quota_api qui sera liée à votre nouvel utilisateur.

Comme dans les précédents travaux, vous devez mettre en œuvre les bonnes pratiques de la programmation objet. Assurez-vous également que votre code est bien bâti et robuste : bonne séparation, évitez les répétitions et gérer les erreurs. Gardez en tête qu'il y aura d'autres parties à ce projet. Il est donc important de bien comprendre chacune des étapes.

À remettre

- Votre solution nettoyée