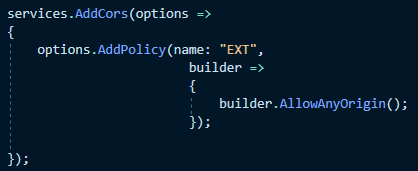
***API qui permet d’effectuer seulement le GET/GET by ID***

Création d’un nouveau cors EXT en renseignant uniquement l’Origin dans le Builder.

AllowAnyOrigin() permet de laissez des requêtes par n’importe quel serveur.

La création s’effectue dans le service du startup.cs.

Ajout du UseCors afin d’utiliser le Cors crée.

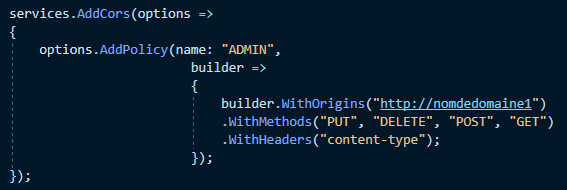
L’ajout de l’appel à cette fonction s’effectue dans la méthode configure du startup.cs.

N’ayant pas renseigné autre chose que l’Origin l’utilisateur pourra uniquement récupérer les informations par le biais du GET.

***API qui permet d’effectuer toutes les requêtes***

Création d’un nouveau cors ADMIN en renseignant :

* Le ou les Origin(s),
* Les méthodes utilisées,
* Le header qui est l’entête de la requête.

La création s’effectue dans le service du startup.cs.

La méthode WithOrigins permet d’ajouter des noms de domaine a la liste blanche (whiteList).

Exemple :

Si vous avez plusieurs noms de domaine il faut l’écrire comme ceci.

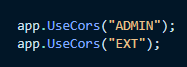




L’utilisateur ayant une de ces Origins pourra effectuer toutes les requêtes possibles.

***Avoir plusieurs Cors actif.***

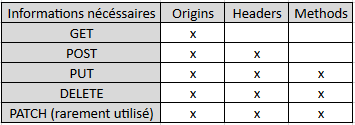
Création de deux cors, l’un qui effectuera uniquement les requêtes GET et l’autre qui effectuera toutes les requêtes.

Ajouter les UseCors.

L’appel des UseCors ont un ordre précis du moins restrictif au plus restrictif afin que les requêtes fonctionnent correctement.

Si le UseCors la plus restrictif est en premier toutes les restrictions s’appliquent sur les Cors qui suit.

***Pour plus d’explications***

Pour comprendre pourquoi on appel certaine méthode lors de la création d’un Cors voici un petit tableau :

Origins :

Afin d’utiliser une API externe en ligne, il faut déterminer les origines de l’appel. Soit on accepte toutes les origines avec le AllowAnyOrigin() qui est fortement déconseillé si on souhaite faire d’autre requête que le GET.

Headers :

Les headers servent à mettre des arguments dans l’entête de la requête.

Exemple : Content-Type, cet argument désigne qu’elle type de données la requête va renvoyer et il est obligatoire pour les requêtes POST, PUT, DELETE, PATCH.

Methods :

Pour les methodes, on pourrait n’en spécifier aucune, mais on peut aussi en spécifier d’autres comme le PUT qui correspond à l’update et le DELETE pour delete en base de données.

**Les requêtes de type http :**

GET :

La requête HTTP GET, est une requête qui permet de récupérer une ou plusieurs données uniquement.

Sans avoir besoin de méthodes ou d’en-tête.

POST / PUT / PATCH:

Les requêtes HTTP POST / HTTP PUT vont permettre de persister des données en base de données.

Il faut donc absolument préciser côté client, l’en-tête de la requête.

Exemple en Javascript :

*(requ étant ma constante et un objet de type XMLHTTPREQUEST)*

DELETE:

La requête DELETE est différente parce que l’API va supprimer des données via un id de l’enregistrement, comme l’id est directement envoyé dans l’url il faut donc spécifier qu’il faut prendre les données de l’url.

Exemple en Javascript :

*(requ étant ma constante et un objet de type XMLHTTPREQUEST)*