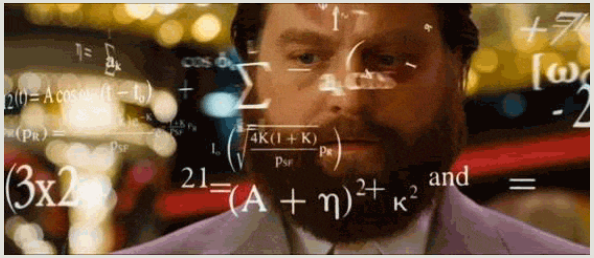
# NestJs


AYMEN SELLAOUTI

# Exercices



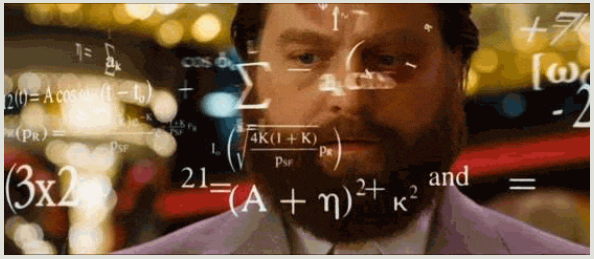
1- Appliquez les contraintes suivantes pour l'ajout d'un Todo.

- La description doit au moins avoir 10 caractères et elle est obligatoire.
- Le name est obligatoire et doit avoir une taille minimale de 3 caractères et une taille maximale de 10 caractères.
- Créer vos propres messages d'erreurs
- Centraliser les dans un même fichier pour optimiser la réutilisation et faciliter la maintenance de votre application.

2- Créer un DTO propre à la méthode de mise à jour. Aucun champs n'est obligatoire pour l'update. Garder les contraintes de la partie ajout (taille).

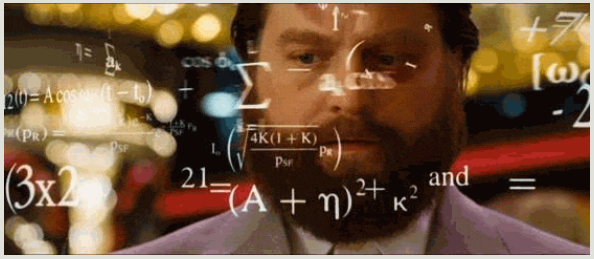
- Le statut doit être une des valeurs de vos status.

# Exercice



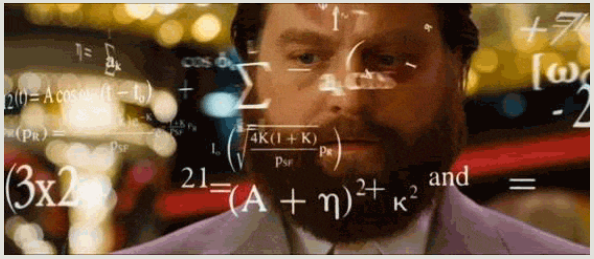

- Configurer TypeOrm au niveau de votre application

# Exercice



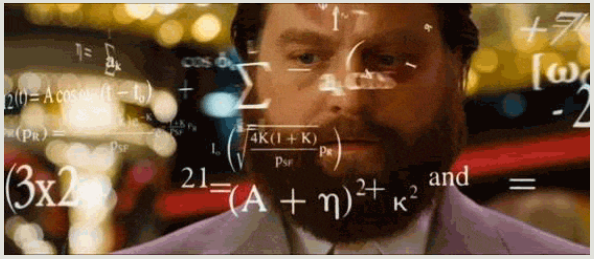
- Créer une entité TodoEntity qui représente votre todo au niveau de la base de données.
- Pour rappel un todo est caractérisé par
  - Id : entier auto incrémenté
  - Name
  - Description
  - CreatedAt
  - Status : de type StatusEnum que vous avez déjà défini

# Exercice



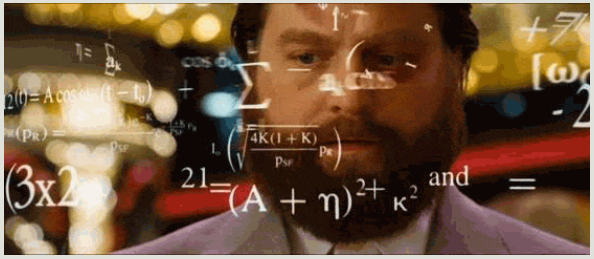
- Mettez à jour votre entité TodoEntity en y ajoutant deux champs updatedAt et deletedAt et faite en sortes que ces deux champs soient automatiquement gérés par TypeOrm.
- Faite la modification nécessaire pour que le champ createdAt ne puisse pas être modifié une fois crée.

# Exercice



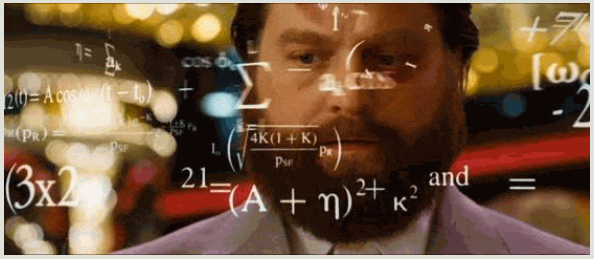

- Etant donné que les champs cretaedAt, updatedAt et deletedAt sont des champs qu'on peut utiliser plusieurs fois, proposer une méthodes pour les rendre réutilisable.

# Exercice



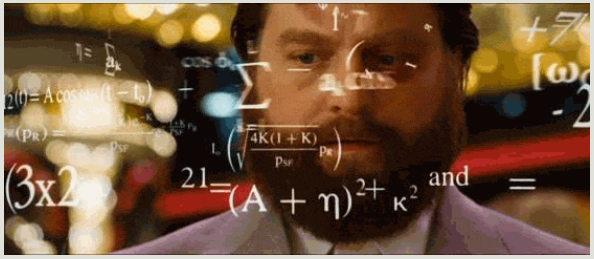
- Modifier la méthode addTodo afin qu'elle puisse ajouter un todo dans la base de données.
- Garder les deux versions de votre code

# Exercice



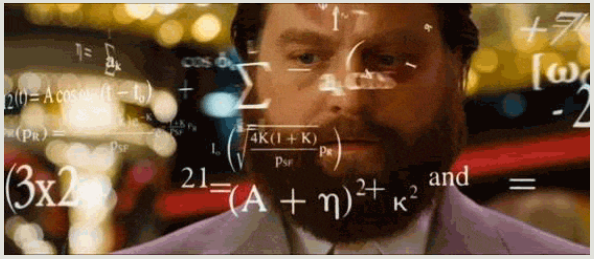
- Créer une nouvelle version de la méthode updateTodo afin qu'elle puisse mettre à jour un todo dans la base de données via son id.

# Exercice



- Modifier la méthode deleteTodo afin qu'elle puisse supprimer un todo dans la base de données via son id.

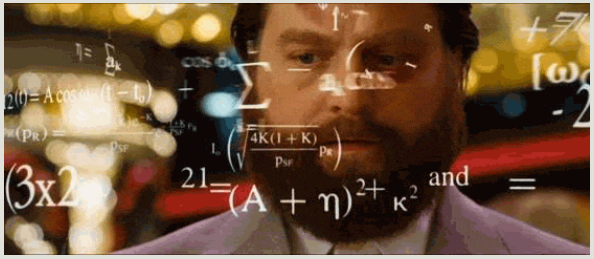
# Exercice




- Modifier la méthode deleteTodo afin qu'elle puisse supprimer un todo dans la base de données via son id d'une façon soft.

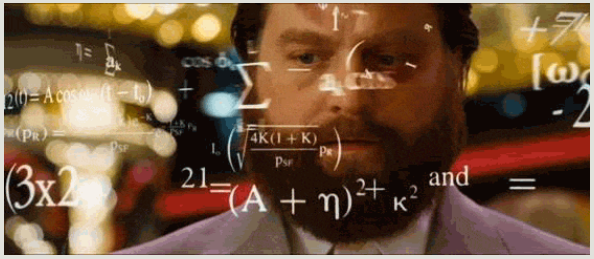
- Implémenter aussi la méthode permettant de restaurer votre todo.

# Exercice



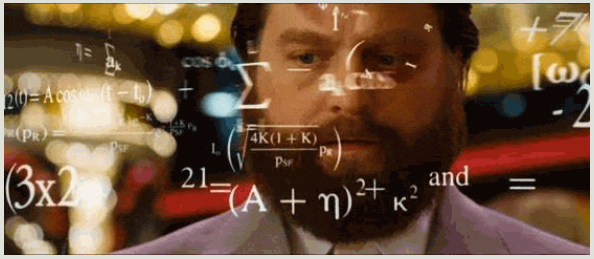

- Préparer une api permettant d'avoir le nombre de todo pour chacun des trois statues

# Exercice



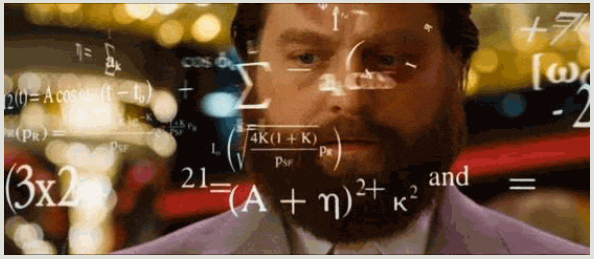

- Faite en sorte d'avoir un endpoint permettant de récupérer l'ensemble des todos.

# Exercice



- Modifier l'api get de sorte qu'il puisse ou non prendre en entrée (via des query parameters) une chaine et un statut et retournant l'ensemble des todos dont la description ou le nom contiennent la chaine **ou** dont le statut est celui recherché.
- Créer un DTO permettant de récupérer le couple statut, critère.

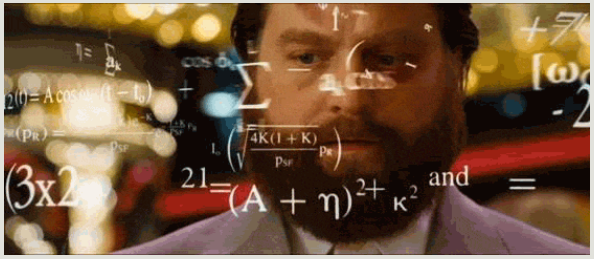
# Exercice




- Créer le endpoint permettant de récupérer un todo par son id.

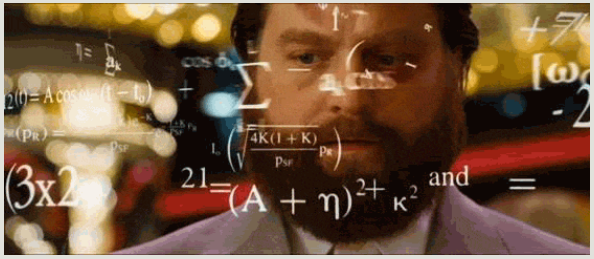
- Gérer le cas ou le todo n'existe pas

# Exercice



- Modifier l'api get de sorte que maintenant on puisse avoir les Todo dont le name **ou** description contiennent la chaine passée en paramètre **et** ayant le statut passé en paramètre.
- Les deux critères de recherche restent optionnels.

# Exercice




- Ajouter le traitement nécessaire pour paginer votre fonction getAll.