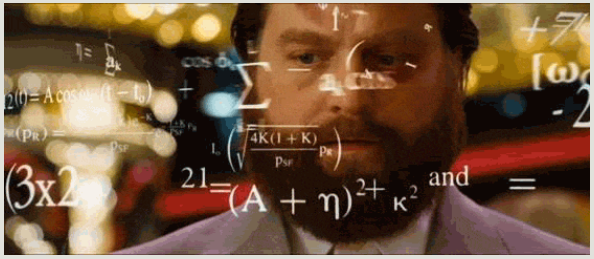
NestJs

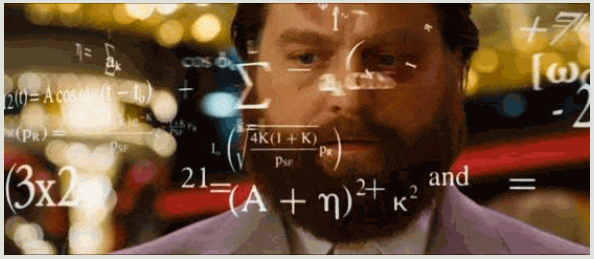
AYMEN SELLAOUTI

Exercice



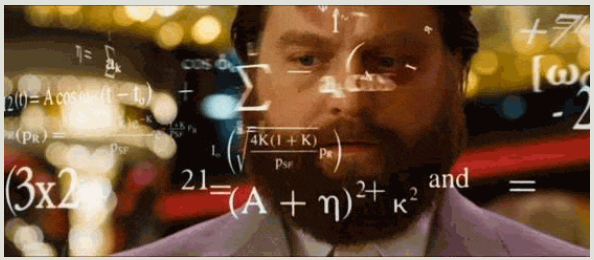
- Créer un module Premier.
- Intégrer le avec votre App.module.ts

Exercice



- Créer un contrôleur premier dans le module premier.
- Faites le nécessaire pour gérer les méthodes GET, POST, DELETE, PUT et PATCH.
- Chaque appel devra uniquement loger le nom de la méthode appelée et la retourner.

Exercice

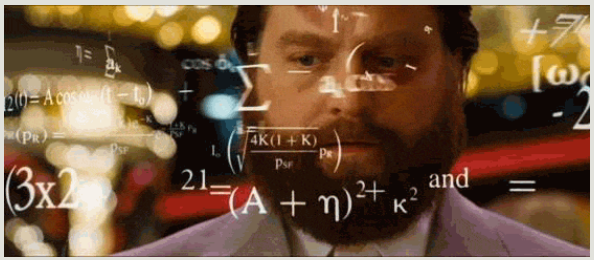


- Créer un module TodoModule.
- Créer un contrôleur TodoController et ajouter le au module TodoModule.
- Créer une Class TodoModel Représentant un Todo qui est caractérisé par son id, son name, sa description, sa date de création et son statut.
- Le statut doit être l'un de ces trois : En attente, En cours, Finalisé. Utiliser un **enum** pour définir ce type.
- Ajouter une méthode Get qui retourne la liste des todos qui est une propriété de votre TodoController.

```
import { Controller, Delete, Get,
Patch, Post, Put } from
'@nestjs/common';
@Controller('todo')
export class TodoController {
  private todos = [];
  @Get()
  getTodos() {
   //  Todo 2 : Get the todo liste
  }
}
```

```
export enum TodoStatusEnum {
  'actif' = "En cours",
  'waiting' = "En attente",
  'done' = "Finalisé"
}
```

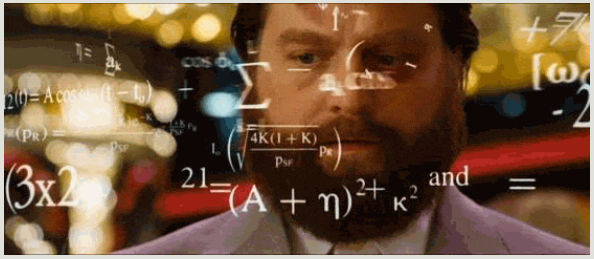
Exercice



- Dans votre contrôleur todo, créer une méthode qui permet d'ajouter un Todo.
- La génération de l'id doit être automatique. Penser à utiliser un générateur d'id unique comme le composant **uuid**.
- La date de création doit aussi être automatique.
- Le statut par défaut est « En attente ».

```
export enum
TodoStatusEnum {
  'actif' = "En cours",
  'waiting' = "En attente",
  'done' = "Finalisé"
}
```

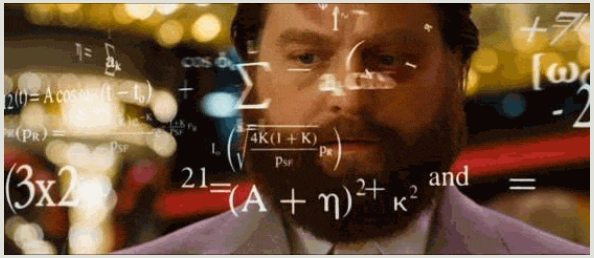
Exercice



Dans votre contrôleur todo créer les méthodes

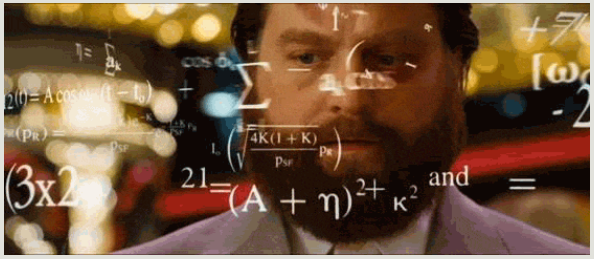
- permettant de récupérer un todo via son id.
- permettant de supprimer un todo via son id.
- permettant de modifier un todo.

Exercice



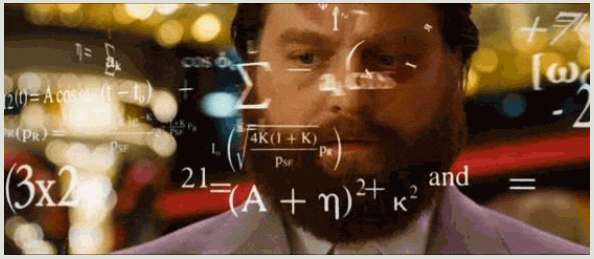
- Ajouter le DTO nécessaire pour ajouter un TODO.
- Modifier votre code en utilisant le DTO.
- Créer un autre DTO pour la mise à jour.

Exercice



- Créer un service todoService permettant de centraliser les fonctionnalités liées au todo.

Exercice



- Créer un module CommonModule
- Ce module va être souvent utilisé pensez à ca en le créant
- Provider la fonction uuid
- Faite en sorte de l'utiliser dans le TodoService