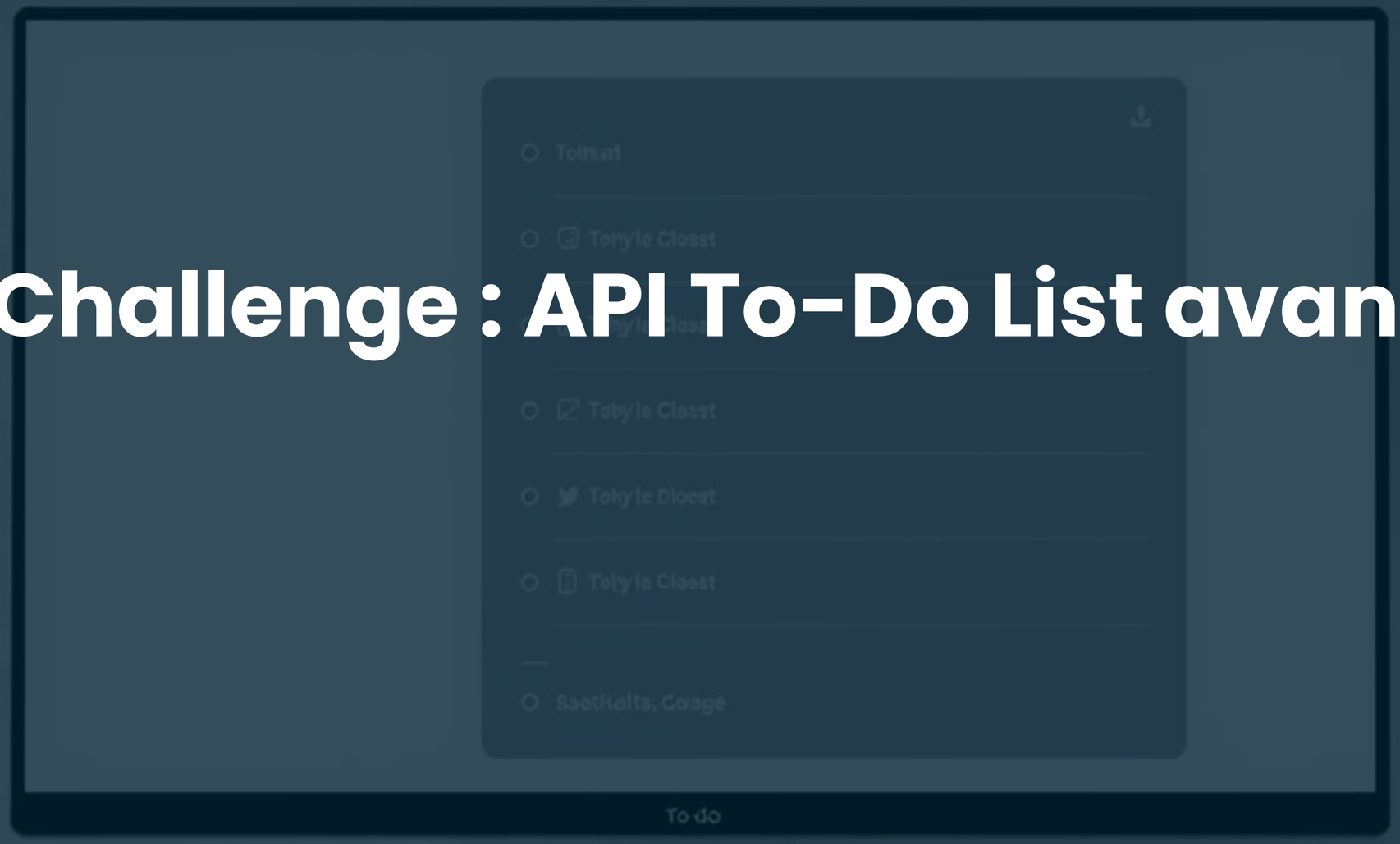
# Code Challenge : API To-Do List avancée



# Architecture et Choix Technologiques

## Objectif :

Développer une API REST de gestion de tâches (To-Do list) avec **NestJS**, permettant de **créer**, **lister**, **mettre à jour**, **supprimer**, et **filtrer** des tâches.Le tout en respectant les **bonnes pratiques de développement backend**.

**Base de Données statique**

**Architecture modulaire**

**Documentation si possible**

# Conception de l’API : Entités et Endpoints

L'entité centrale de notre API est la **Tâche** (Todos), définie par un identifiant unique, une description, un statut d'achèvement (booléen), et des horodatages de création et de dernière mise à jour. Ces champs garantissent la complétude et la traçabilité de chaque tâche.

Les endpoints suivent les conventions **RESTful**, assurant une interface claire et prédictible pour les opérations CRUD et le filtrage. La validation des données est gérée par les Pipes de NestJS, notamment via la bibliothèque **class-validator**, garantissant l'intégrité des informations reçues par l'API. La gestion des erreurs est standardisée grâce aux exceptions HTTP, offrant des retours compréhensibles en cas de problème.

| | |
|---|---|
| POST | /todos |
| GET | /todos |
| GET | /todos/:id |
| PATCH | /todos/:id |
| DELETE | /todos/:id |

# 🛠️ Fonctionnalités attendues :

## 🔹 CRUD de base

- POST /todos : Créer une tâche avec les champs suivants :
  - title (obligatoire)
  - description (optionnel)
  - priority (low, medium, high) avec valeur par défaut medium
  - tags (tableau de chaînes, optionnel)
  - isFavorite (booléen, optionnel, défaut : false)
- GET /todos : Retourner toutes les tâches
- GET /todos/:id : Retourner une tâche par ID
- PATCH /todos/:id : Mettre à jour tout ou partie d'une tâche
- DELETE /todos/:id : Supprimer une tâche

## 🔹 Filtres avancés (sur la route GET /todos)

- Filtrer par **priorité** : ?priority=high
- Filtrer par **tag** : ?tag=travail
- Filtrer par **favoris** : ?isFavorite=true
- (Bonus) Ajouter **recherche par mot-clé** dans le titre ou la description : ?search=urgent

# 💡 Contraintes techniques

- Utiliser **NestJS** (architecture modulaire recommandée)
- Utiliser **DTOs** + **Validation (class-validator)** pour toutes les entrées
- Utiliser **services injectables** pour la logique métier
- Utiliser **tableau en mémoire** comme base de données (pas besoin de BDD réelle pour ce challenge)

# Bonne chanche