# Application Angular TODO List

## Objectif

Voici les attributs d'une tâche avec les types associés :
Le but de cette application sera de gérer une liste d'actions à faire.  C'est-à-dire : créer, modifier, supprimer des tâches.
De plus, chaque tâche à faire sera affectée à une personne.

On va considérer que le type qui sert à manipuler les tâches à faire s'appelle Todo. De même, on va considérer que le type qui sert à manipuler les personnes sera Person.

Cette application aura besoin de manipuler des données. Ces données seront mockées dans le front, ce qui signifie que vous ne devez pas (ou n'avez pas besoin) implémenter une API.
Le mock est suffisant pour l'exercice.

Voici les attributs d'une tâche avec les types associés :
**titre : string**
**person: Person**
**startDate: Date**
**endDate: Date**
**priority : Enum**
**labels: Enum[]**
**description: string**

Les attributs **priority (Facile, Moyen, Difficile)** et **label (HTML, CSS, NODE JS, JQUERY)**, comme vous l'avez compris sont des énumérations dont les valeurs sont spécifiées entre parenthèses. Tout comme l'attribut person, leur valeur sera choisie via une liste déroulante.

On fera simple pour les attributs de **Person** : **name, email, phone**. Tous de type string.

Nous aurons donc deux modules à gérer. Qui vont visuellement se ressembler :

- Une liste
- Et pour chaque élément de la liste, une modale qui va s'afficher pour le détail

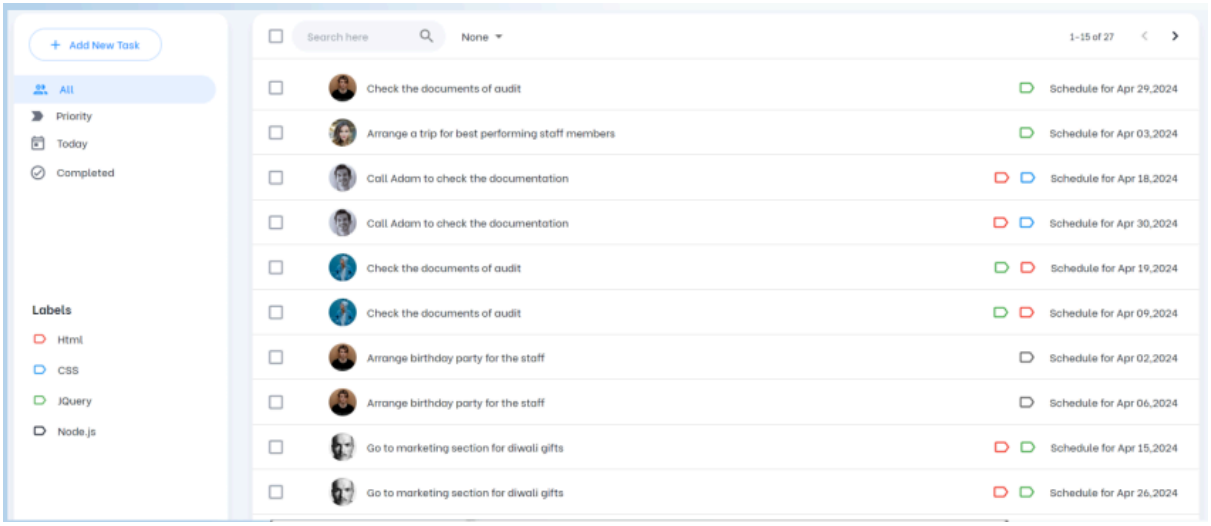
Pour l'implémentation des appels API, utiliser **HttpClient**.
Pour l'implémentation des mocks, utiliser la librairie **json-server**

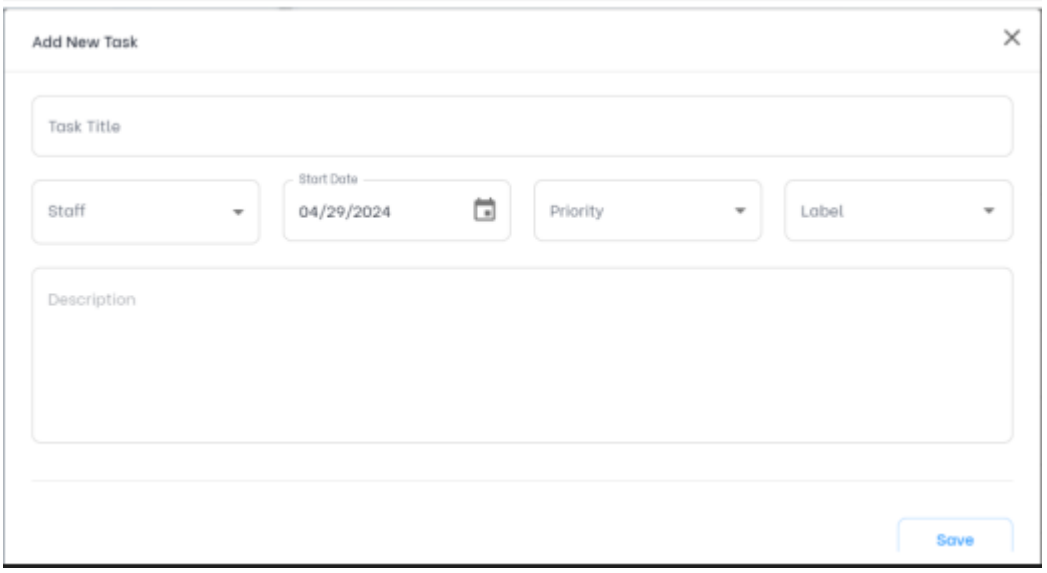
Les listes (des tâches ou des personnes) proposera des boutons pour :

- Ajouter une tâche (respectivement une personne)
- Supprimer une tâche (respectivement une personne)
- Éditer une tâche (respectivement une personne)

Filtrer le contenu en fonction de certains critères (priorité et label)
Par ailleurs, il faudra implémenter un système de pagination.
Pour implémenter le tableau, vous allez utiliser le composant [ng2-smart-table]

Exemple de maquette visuelle de la liste des tâches ainsi que de la modale (s'en inspirer pour faire celle des personnes)




# Instructions

Voici les fonctionnalités à implémenter et les règles de gestion :
1. Faire passer le tableau en lecture seule.

Pour ajouter une nouvelle tâche, il faudra :

1. Cliquer sur Ajouter
2. Une nouvelle fenêtre modale s'affiche
3. Les informations sont rentrées dans cette fenêtre modale
4. Lorsqu'on sauvegarde, une ligne est rajoutée sur le tableau

2. La même fenêtre modale sera utilisée pour l'ajout ou l'édition d'une tâche

3. Règles de validation dans la modale d'ajout/édition

1. Le nom d'une personne doit avoir au moins 3 caractères après avoir été trim()
2. Le titre d'une tâche doit avoir au moins 3 caractères après avoir été trim()
3. L'email d'une personne doit être valide
4. Le nom d'une personne doit être unique dans le tableau de la liste des personnes
5. Une tâche peut avoir plusieurs labels
6. Lorsqu'on indique une tâche comme terminée, l'attribut endDate se met automatiquement et il n'est plus modifiable

4. Si une des règles de validation de la modale échoue, le champ concerné du module concerné doit être mis en rouge avec un message d'erreur explicite.

5. On ne peut pas enregistrer les informations de la modale s'il y a des erreurs

6. Dans la modale des tâches, le choix de personne se fait via un composant **Autocomplete**.

7. Il faudra bien penser à implémenter les filtres pour les listes. Des exemples de filtres pour la liste des tâches sont proposés dans la slide précédente. Il faudra en proposer pareil pour les personnes.

**BONUS 1 (non obligatoire)** : implémenter l'internationalisation. C'est-à-dire que vous rajouter une liste déroulante avec comme éléments Français et Anglais. Le choix d'une langue traduit
instantanément l'ensemble de la page sans avoir besoin de la rafraîchir. Utiliser la librairie transloco pour cela.

**BONUS 2 (non obligatoire)** : implémenter l'export Excel et PDF. On pourra exporter des tâches actuellement affichées à l'écran au format Excel ou PDF. Il peut s'agir de toutes les tâches ou des tâches après l'application d'un filtre.

# Règles spécifiques concernant le code

Plusieurs règles seront à respecter concernant le code.

- Le code doit être fait en **TypeScript** (pas en Javascript, mais en TypeScript)
- On utilisera **Angular**. Pas React ou VueJS, mais bien Angular.

- A chaque étape de votre progression, il faudra faire un commit. Dans l'analyse de votre solution, nous regarderons la liste des commits et leur contenu. Concrètement, nous allons voir l'évolution du code.

- Le résultat du code final sera pusher sur Github dans un dépôt public dont vous communiquerez le lien.

- Vous aurez en revanche le loisir d'utiliser la bibliothèque de votre choix pour le bonus 2.

- Une attention particulière sera portée sur le nom de vos variables. Elles doivent avoir du sens et permettre une lecture et une compréhension facile de votre code.

- Pour le style vous utiliserez tailwindcss et Angular Material