Projet Angular

L'objectif de ce projet est de vous permettre de vous renforcer sur les notions que l'on a pu voir en cours. Le projet est à réaliser en **binôme**. Si vous ne trouvez pas de binôme, merci de nous le préciser pour que nous puissions former un trinôme. Il est inutile de nous demander à l'avance si vous pouvez former un trinôme, celui-ci sera formé si nécessaire par vos enseignants.

Vous pouvez d'ores et déjà vous inscrire lorsque votre binôme est formé :
https://forms.office.com/e/Zm8ZEWf1vw

La date de rendue sera aux alentours du **10/02/2023** (date qui sera confirmée dès que possible).

Dans la mesure où ce projet sera à réaliser en binôme et que certains d'entre vous pourront partir à l'étranger au semestre 4, nous vous demandons de mettre en place un système de versioning au travers de l'utilisation de GitHub.
Vous devrez proposer de manière régulière des versions de votre site sur GitHub.
Étant donné que la plupart d'entre vous n'ont pas encore pu utiliser ce système de versioning, la bonne utilisation de GitHub sera valorisée sous forme de points bonus. Toutefois, l'utilisation de GitHub même sommaire sera obligatoire et pénalisée si vous ne l'utilisez pas. Nous ne récupérerons vos sources que par ce moyen.

Ce projet se découpe en 4 étapes avec un niveau croissant de difficulté. Lors de votre rendu de ce projet, vous devrez spécifier à quelle étape vous vous êtes arrêté et proposer des versions de votre code sur GitHub pour chacune des étapes.

Comme point de départ de ce projet, vous devez partir de vos réalisations du TP4, que ce soit pour l'API REST ou pour la partie Angular. Si vous n'êtes pas arrivé à terminer ces 2 TPS, vous pouvez partir de la correction que nous avons mise à disposition : https://github.com/AmauryAug/TP4Correction

Pour mieux appréhender Git et GitHub de nombreux tutos existent en ligne. Vous pouvez apprendre les notions de base et vous entraîner sur ce site si vous le souhaitez :

https://skills.github.com/

https://learngitbranching.js.org/

Étape 1 Ajout d'un statut de tache

Une tache a maintenant un nouvel attribut : statut qui peut avoir 4 valeurs : Absence de valeur, En attente, En cours, Terminé.

Modifier le fichier tache.ts pour prendre en compte cette modification.

Vous devez maintenant modifier votre fichier taches.component.html pour afficher les taches dans 4 colonnes. Voici à quoi doit ressembler votre page finale :

À noter :

- Si j'ajoute une tache pour un statut particulier, il faut que la tache soit enregistrée dans la base de données avec le bon statut !
- Le CSS ne sera pas noté pour ce projet, mais il faut tout de même que le rendu soit un minimum lisible et ergonomique.

Étape 2 mise en place du Drag & drop

Dans cette étape l'objectif est de pouvoir déplacer une tache d'une liste à une autre avec du Drag&Drop.

Nous ne fournirons pas d'instruction détaillée sur comment mettre en place ce mécanisme pour éprouver votre capacité de recherche.

Cependant, voici 2 liens :

- La documentation officielle d'une librairie permettant de réalisation le Drag & Drop : https://material.angular.io/cdk/drag-drop/overview
- Un lien vers un tutoriel qui vous explique comment mettre en œuvre : https://www.freakyjolly.com/angular-material-drag-and-drop-across-multi-lists-example/

À noter :

- Dans le projet d'API REST, il faudra modifier la fonction tachePut pour pouvoir mettre à jour le nouveau champ statut.
- Dans la callBack de dépôt d'élément (fonction drop() dans le tutoriel) il faudra :
    - Modifier le statut de la tache qui a été déplacé.
    - Et appeler le webService de mise à jour de la tache (tachePut).
- Nous ne vous demandons pas de gérer la mise à jour de l'ordre des taches dans ce projet.
  Si vous modifiez l'ordre avec le drag & Drop et que vous rafraîchissez votre page, la modification ne sera pas prise en compte.

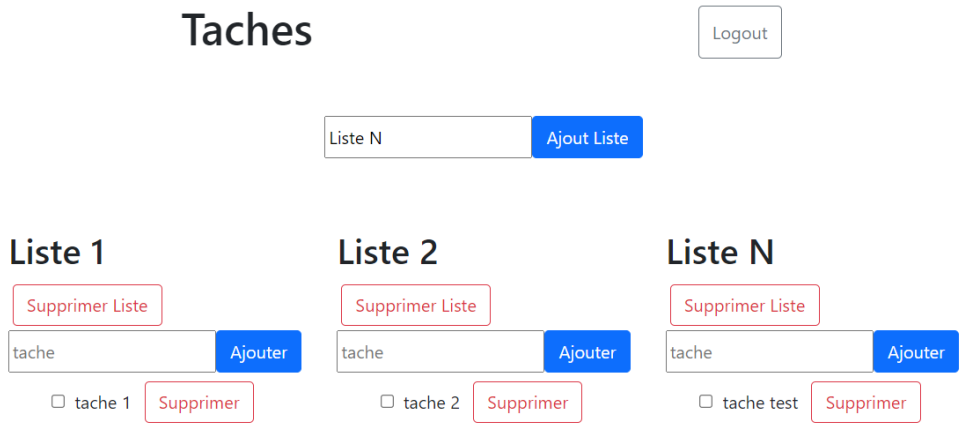
Étape 3 liste dynamique

Nous voulons maintenant ne plus avoir de statut fixe, mais pouvoir dynamiquement ajouter de nouveau type de liste de tâche.

Pour réaliser cette partie, il faut :

- Ajouter un bouton permettant d'ajouter des listes
- Quand une liste est présente en base, il faut afficher toutes les tâches qui sont assignées à cette liste (vous pouvez faire un nouveau webService qui récupère toutes les taches d'une liste).
- Il faut maintenant pouvoir supprimer cette liste avec toutes les taches qui la composent.
- Il vous faudra également modifier votre code pour que le drag & Drop continus de fonctionner entre listes en rendant dynamique la directive cdkDropListeConnectedTo.

Voici un exemple de ce à quoi l'application pourrait ressembler à la fin de cette étape.



Étape 4 Identification

Votre Todo List étant maintenant entièrement fonctionnel il faut la sécuriser. Afin d'y parvenir, vous devez proposer un système de connexion permettant à un utilisateur de n'accéder qu'à sa Todo List.

La mise en place de cette solution devra vous faire repenser la structure de vos BDD, faites attention à créer des versions dans GitHub pour ne pas perdre les travaux déjà réalisés.

La réalisation de cette partie peut être vue en plusieurs étapes :

- Plusieurs utilisateurs existent en BDD (créé à la main ou par formulaire d'inscription), seule la connexion vers l'espace de l'utilisateur est fonctionnelle.
- Les utilisateurs peuvent s'inscrire sur le site à l'aide d'une page d'inscription et peuvent ainsi accéder au site.
- La page d'inscription avertit l'utilisateur qui ne peut pas utiliser certains pseudos si ces derniers existent déjà en base de données.

Le système d'authentification et d'identification pouvant être réalisée de manière indépendante, il est recommandé de faire des versions pour chaque élément implémenté. En procédant de cette manière vous vous assurerez d'avoir tous les éléments fonctionnels même si vous ne parvenez pas à tout intégrer.