POLYTECH MARSEILLE Aix*Marseille Université **Informatique – 4A**

Applications web et mobiles

TP n°15 : La page des topics du forum Ionic

Étape 72 – La page des topics

Vous allez maintenant vous attaquer à la « vue » des topics : dans votre page des topics, écrivez le code qui permet d'afficher la liste des topics ainsi que le menu déroulant qui permet de les trier. Ici, il s'agit essentiellement d'un copier/coller de ce que vous avez déjà écrit pour les cours ainsi que d'une partie du TypeScript de votre code *Angular*. La seule exception réside dans les tris, qui se font : par topic, par date de dernier post et par nombre de posts.

Étape 73 – Rajout du back button

Rajoutez un « *back button* » dans la « toolbar » de la page des topics.

Étape 74 – Le bouton d'ajout de nouveaux topics

Rajoutez en haut de la page des topics un bouton permettant de rajouter de nouveaux topics. Reliez-le à une « alerte », similaire à celle que vous aviez construite pour le composant de login. Pour l'instant, dans l'alerte, affichez juste un texte « bidon » et faites en sorte qu'il y ait 2 boutons en bas de l'alerte : « cancel » et « créer ». Testez que votre alerte s'affiche bien.

Étape 75 – Les rôles des boutons « cancel » et « créer » – partie 1

Dans votre TypeScript, la méthode `onDidDismiss()` de votre alerte vous renvoie une `Promise` qui, une fois résolue, contiendra des informations, notamment, lequel des deux boutons « cancel » ou « créer » a fermé l'alerte. Pour visualiser ces informations, après le `alert.present()`, tapez l'instruction :

```
alert.onDidDismiss().then((data) => {console.log(data);});
```

Regardez ce que cela produit dans la console quand vous fermez votre alerte en appuyant le bouton « cancel », puis le bouton « créer ».

Étape 76 – Les rôles des boutons « cancel » et « créer » – partie 2

Il serait utile pour la suite que le rôle du bouton « créer » soit `créer` plutôt que `undefined`. Pour cela, dans le tableau `buttons` du `alertController.create`, au lieu de spécifier uniquement les noms des boutons, vous allez indiquer plus d'informations : chaque bouton sera défini par un objet Javascript :

```
{ text: 'nom du bouton', role: 'nom du bouton' }
```

Regardez maintenant ce que cela donne dans la console quand vous cliquez sur les boutons « cancel » et « créer ».

Étape 77 – Le nom du nouveau topic

Comme vous l'avez vu en cours, vous avez créé votre alerte en passant au `alertController.create` un objet avec un champ `message`. Pour ajouter une balise `<input>`, il suffit de remplacer ce champ par un champ :

```
inputs: [
  {
    name: "l'attribut name de la balise <input>",
    type: "l'attribut text de la balise <input>",
    placeholder: "l'attribut placeholder de la balise <input>"
  }
]
```

Ajoutez donc votre balise `<input>` et testez ce que cela donne. Notamment, regardez dans la console comment récupérer ce que vous avez tapé dans l'*input* (cf. la méthode `onDidDismiss`).

Étape 78 – La fin de l'alerte

Maintenant, vous savez quel bouton a été utilisé pour fermer l'alerte ainsi que le texte qui a été saisi dans l'*input* de l'alerte. Créez une nouvelle méthode prenant en paramètre ce texte et qui va le transmettre au *backend* afin de créer le nouveau *topic*. Pour cela, inspirez-vous de ce que vous aviez écrit dans votre forum *Angular* dans le composant de dialogue du « create topic », ainsi que dans le composant des topics (mise à jour de la liste des topics à afficher).

Testez bien que votre base de données est bien mise à jour si le sujet n'existe pas déjà. Lorsque le sujet existe, affichez une alerte indiquant que ce sujet existe déjà. Vérifiez bien également que votre nouveau sujet s'affiche dans la page des topics.

Étape 79 – Le back button

Maintenant que votre page des *topics* est correcte, comptez le nombre de *topics* qu'elle contient et appuyez sur le *back button*. Si vous n'avez pas de *back button* (ce qui peut arriver si vous avez rechargé manuellement votre page dans votre *navigateur*), retournez à la page de *login*, sélectionnez un cours, ajoutez un nouveau *topic*, et recomptez le nombre de *topics* dans votre page, puis appuyez sur le *back button*. Observez le nombre de topics indiqué dans le cours que vous venez de modifier. Normalement,

le nombre de topics indiqué dans la page des cours est incorrect. Cela vient du fait que, lorsque l'on clique sur le *back button*, la méthode `ngOnInit` n'est pas réexécutée. Or, *a priori*, c'est dans celle-ci que vous avez rechargé du *backend* la liste des cours. Pour pallier cela, placez le contenu de votre `ngOnInit` dans la méthode `ionViewWillEnter`, comme vu en cours. Retestez.

Optionnel : pour ceux/celles qui souhaitent aller plus loin

Étape 80 – La page des posts

La page des *posts* est essentiellement un copier/coller de la page des *topics* de votre forum *Ionic* ainsi que des `sendMessage` de votre forum *Angular*. Contrairement au forum *Angular*, pour un smartphone, on ne va pas utiliser un éditeur complexe (*rich editor*) pour ajouter des posts, mais plutôt utiliser le même `<input>` que pour ajouter des *topics*. À noter ici que vous pouvez utiliser des balises `<ion-card></ion-card>` plutôt que des `<ion-label></ion-label>` pour l'affichage du contenu des *posts*. Cela donne un rendu plus sympathique.