Projet - WebMobile (petits gazouillis)
420-P5C
20% de la note finale

Description

Créer un projet web mobile offrant presque les mêmes fonctionnalités que celles offertes par le projet `petit gazouillis`. Ce projet comprend une application web mobile développée dans un environnement `expo/React Native` et un serveur `REST API` développé à partir du projet `petits gazouillis`. L'application web mobile devra fonctionner en version `web`. De plus, l'interface de l'application web mobile devra permettre l'actualisation quand il y a une nouvelle publication survient.

- Ce projet se fait seul ou en équipe de deux . SVP confirmer par courriel les deux noms de l'équipe. Je me réserve le droit de ne pas valider l'équipe.
- Présentation de l'énoncé : 23 octobre 2020
- La date de remise est lundi 7 décembre 2020 avant 23H59 (- 10% par jour à partir de 23h59) .
- Vous devrez remettre votre projet `petits_gazouillis_par_votre_nom.zip` sur LÉA contenant une application `React Native` dans un dossier `web_mobile` et un serveur `REST API` dans un dossier `serveur_rest_api`.
- Afin de faciliter la correction, vous devrez lors d'une rencontre Zoom démontrer que votre projet respecte les critères de correction (durant la semaine 15 et 16)
- La correction du projet se fera à partir de votre projet (partage d'écran).
- Puisque ce projet est réalisé pour la première fois, il est possible que des modifications y soient apportées en cours de route.
- L'exploration est un des objectifs de ce projet. Vous devrez démontrer une très grande autonomie dans la réalisation du projet. Cependant, sur demande de votre part, je pourrai produire des capsules vidéo sur des aspects spécifiques.

Par où débuter? Je vous suggère ce tutoriel https://docs.expo.io/ (`Slow start`)

PARTIE 1- (35%) Développer un serveur `REST API`
En se basant sur le projet `petits gazouillis`, ce serveur aura les caractéristiques minimales suivantes:

A => Gérer les jeton
(5%) `/api/jeton`
`POST` - demander un jeton avec `utilisateur:mot de passe`

B => Gérer les utilisateurs
(7.5%) `/api/utilisateur`
`GET` - Lister les informations d'un utilisateur (`<id>`) avec `jeton`
`GET` - Lister les informations des utilisateurs avec `jeton`
`GET` - Suivre un utilisateur avec `jeton spécifique`
`GET` - Ne plus suivre un utilisateur avec `jeton spécifique`

C => Gérer les publications
(7.5%) `/api/publications`
`GET` - Lister les informations d'une publication (`<id>`) avec `jeton`
`GET` - Lister les informations des publications avec un `jeton`
`POST` - Créer une publication avec un `jeton spécifique`

* `jeton spécifique` appartient à un `utilisateur spécifique`. C'est à vous de déterminer comment réaliser ceci.

D => Gérer les erreurs
(5%) Gestion de tous les messages d'erreurs par des flux `JSON` contenant les messages d'erreurs.

E => Gérer les événements demandant une actualisation de l'interface utilisateur.
(10%) Les utilisateurs sont informés de nouvelles publications par des messages transmis par `websocket`.

Note : Vous êtes libre d'ajouter toutes fonctionnalités que vous jugerez nécessaires à la réalisation du projet.

PARTIE 2- (65%) Développer une application web mobile

- Vous êtes libre de déterminer le format de l'interface.
- L'application web mobile est obligatoirement un projet `expo REACT NATIVE`
- L'application web mobile doit obligatoirement fonctionner en version `Web`–

Votre application web mobile devra avoir les caractéristiques suivantes:

A => Actualisation
(20%) Lors de la réception d'un événement de type `nouvelle publication`, un bouton est activé afin de permettre l'actualisation de la liste des publications.

B => Fonctionnalités
(40%) Votre application devra offrir les fonctionnalités ci-dessous. C'est à vous de déterminer de quelles manières ces fonctionnalités seront offertes à l'utilisateur par l'intermédiaire de l'interface.

1. Établir une session
2. Terminer un session
3. Publier un message
4. "Suivre" ou "ne plus suivre" un utilisateur
5. Afficher les listes sur plusieurs `page` et `suivant, précédent`
6. Filtrer la liste des messages(filtrer: mes messages, messages suivis, tous les messages, etc.)

C => Erreurs
(5%) Gérer avec élégance les erreurs

En cas d'erreur, l'interface réagit avec "élégance". Voici une liste d'erreurs auxquelles votre interface doit réagir.

1. établir une session: utilisateur ou mot de passe

Voilà!