ENVOYER DES NOTIFICATIONS AVEC EXPO

PRÉREQUIS

- */!\ Attention, si des essais infructueux ont été faits, ça peut valoir le coup de désinstaller/réinstaller Expo Go sur le téléphone avant de tester une nouvelle version*

- Avoir construit une app avec expo
  - cd dans notre projet
    npx create-expo-app NomDeLapp
  - cd NomDeLapp
- Avoir installé eas-cli
  npm install -g eas-cli
- Avoir testé que l'app fonctionne déjà comme ça
  npm start
- Avoir fait un build (cela génère un id d'application et une clé Android)
  eas build
  [patienter un peu le temps que ça charge]
  Mémo testé en sélectionnant uniquement Android au moment du build
- Avoir installé expo-notifications
  npx expo install expo-notifications

DANS APP.JS (EN HAUT DU FICHIER)

- J'importe les hooks dont je vais avoir besoin :

```
import { useState, useEffect, useRef } from 'react';
```

- J'importe des éléments dont j'aurai besoin pour les notifs (expo-notifications)

```
import * as Notifications from 'expo-notifications';
```

- Je n'oublie pas d'importer les composants basiques de ReactNative utiles à mon app.

- Je paramètre l'apparition des notifications (alerte / son / etc) :

```
Notifications.setNotificationHandler({
  handleNotification: async () => ({
    shouldShowAlert: true,
    shouldPlaySound: false,
    shouldSetBadge: false,
  }),
});
```

DANS APP.JS (AU DEBUT DU COMPOSANT APP())

- J'utilise useState et useRef pour paramétrer les notifs, le jeton et les écouteurs de notifs
- Si j'ai bien compris, c'est cet écouteur de notification qui fait que l'on peut recevoir une notification même quand l'app n'est pas active.

```
export default function App() {
  const [expoPushToken, setExpoPushToken] = useState('');
  const [notification, setNotification] = useState(false);
  const notificationListener = useRef();
  const responseListener = useRef();
```

- En-dessous de ça, je définis une fonction qui va programmer la notification

```
async function schedulePushNotification() {
  await Notifications.scheduleNotificationAsync({
    content: {
      title: '🙂 Coucou',
      body: 'Ca fait plaisir de recevoir une notification souriante !',
    },
    trigger: { seconds: 2 },
  });
}
```

- En-dessous, je gère les autorisations et le jeton dans une fonction asynchrone
  Cette fonction renvoie le jeton… Voir page suivante

DANS APP.JS (TOUJOURS AVANT LE RETURN)

FONCTION GERANT LES PERMISSIONS ET LE JETON

/!\ Je ne passe plus par expo-permissions car c'est "deprecated"

```
async function registerForPushNotificationsAsync() {
  let token;
    const { status: existingStatus } = await Notifications.getPermissionsAsync();
    let finalStatus = existingStatus;

    if (existingStatus !== 'granted') {
      const { status } = await Notifications.requestPermissionsAsync();
      finalStatus = status;
    }

    if (finalStatus !== 'granted') {
      Alert.alert(
        'No Notification Permission',
        'please goto setting and on notification permission manual',
        [
          { text: 'cancel', onPress: () => console.log('cancel') },
          { text: 'Allow', onPress: () => Linking.openURL('app-settings:') },
        ],
        { cancelable: false },
      );
      return;
    }

    token = (await Notifications.getExpoPushTokenAsync()).data;

    if (Platform.OS === 'android') {
      Notifications.setNotificationChannelAsync('default', {
        name: 'default',
        importance: Notifications.AndroidImportance.MAX,
        vibrationPattern: [0, 250, 250, 250],
        lightColor: '#FF231F7C',
      });
  }

  return token;
}
```

Dans APP.JS (toujours avant le return)

UseEffect pour gerer les permissions (et le jeton)

- La partie avant le return configure les écouteurs de notifications et de leurs réponses.
- La partie dans le return est une fonction de nettoyage qui sera exécutée quand le composant est démonté (éviter les fuites mémoire). On supprime les écouteurs configurés précédemment.

```
useEffect(() => {
  registerForPushNotificationsAsync().then((token) => setExpoPushToken(token));

  notificationListener.current =
Notifications.addNotificationReceivedListener((notification) => {
    setNotification(notification);
  });

  responseListener.current =
Notifications.addNotificationResponseReceivedListener((response) => {
    console.log(response);
  });

  return () => {
    Notifications.removeNotificationSubscription(notificationListener);
    Notifications.removeNotificationSubscription(responseListener);
  };
}, []);
```

UseEffect pour envoyer une notification

Ici on envoie une notification basique au démarrage de l'app. On peut aussi surveiller une variable grâce au useEffect et déclencher une notification quand la valeur de cette variable change (par exemple la température).

```
useEffect(() => {
  schedulePushNotification();
}, []);
```

Dans le return du composant APP

Il n'y a rien besoin de faire en particulier. Eventuellement ajouter un bouton si besoin de déclencher la notification (ici ce n'est pas le cas).

Si j'ai encore des erreurs ou des warnings

- Je fais un **npm install** pour m'assurer que tout est installé correctement
- Je désinstalle puis réinstalle l'application Expo Go.