TP4 – Navigation entre écrans avec React Navigation

Introduction

Dans une application mobile, il est essentiel de pouvoir naviguer entre différents écrans. Une interface composée d'une seule vue limite fortement l'expérience utilisateur : pas de profil, pas d'espace paramètre, pas de pages détaillées, etc.
Pour répondre à ce besoin, React Native s'appuie sur une bibliothèque officielle, largement adoptée et très flexible : **React Navigation**.

Objectif du TP : la mise en place d'un système de navigation entre plusieurs écrans, l'envoi de paramètres et la personnalisation du header.

1. Installation et configuration de React Navigation

React Native ne propose pas de système de navigation intégré.
La bibliothèque recommandée est **React Navigation**, utilisée dans la majorité des applications React Native.

Site officiel : https://reactnavigation.org/

Étapes d'installation

Dans un projet Expo déjà créé, exécutez les commandes suivantes dans le terminal :

```
npm install @react-navigation/native
npx expo install react-native-screens react-native-safe-area-context
npm install @react-navigation/native-stack
```

- **@react-navigation/native** : constitue le cœur du système de navigation.
- **react-native-screens** et **react-native-safe-area-context** : optimisent la gestion des écrans sur iOS et Android.
- **@react-navigation/native-stack** : permet d'utiliser la navigation empilée (stack navigation), identique au fonctionnement des écrans sur un téléphone.

2. Configuration du conteneur de navigation

Créez un fichier **App.js** et insérez la structure de navigation suivante :

```
import * as React from 'react';
import { NavigationContainer } from '@react-navigation/native';
import { createNativeStackNavigator } from '@react-navigation/native-stack';
import Accueil from './screens/Accueil';
import Profil from './screens/Profil';

const Stack = createNativeStackNavigator();

export default function App() {
 return (
  <NavigationContainer>
   <Stack.Navigator initialRouteName="Accueil">
    <Stack.Screen name="Accueil" component={Accueil} />
    <Stack.Screen name="Profil" component={Profil} />
   </Stack.Navigator>
  </NavigationContainer>
 );
}
```

- **NavigationContainer** : composant racine indispensable, responsable de la gestion globale de la navigation.
- **createNativeStackNavigator()** : génère une pile d'écrans (comme dans un mobile Android/iOS).
- **initialRouteName** : définit l'écran affiché au démarrage.

3. Création de l'écran « Accueil »

Dans un dossier `screens`, créez le fichier **Accueil.js** :

```
import React from 'react';
import { View, Text, Button, StyleSheet } from 'react-native';

export default function Accueil({ navigation }) {
 return (
  <View style={styles.container}>
   <Text style={styles.title}>Bienvenue sur l'écran d'accueil 🎉</Text>
   <Text style={styles.text}>Appuyez sur le bouton pour aller au profil.</Text>

   <Button
```

```
      title="Aller au profil"
      onPress={() => navigation.navigate('Profil', { nom: 'sarah' })}
    />
  </View>
 );
}

const styles = StyleSheet.create({
 container: {
  flex: 1,
  justifyContent: 'center',
  alignItems: 'center',
  backgroundColor: '#eaf4ff',
 },
 title: {
  fontSize: 24,
  fontWeight: 'bold',
  marginBottom: 15,
  color: '#333',
 },
 text: {
  fontSize: 16,
  marginBottom: 20,
  color: '#555',
 },
});
```

`navigation.navigate('Profil', { nom: 'sarah' })`

➞ Permet d'aller vers l'écran **Profil**, tout en envoyant un paramètre, ici `nom`.

4. Création de l'écran « Profil »

Fichier **Profil.js** :

```
import React from 'react';
import { View, Text, Button, StyleSheet } from 'react-native';

export default function Profil({ route, navigation }) {
 const { nom } = route.params;

 return (
  <View style={styles.container}>
   <Text style={styles.title}>Profil de {nom}</Text>
   <Text style={styles.subtitle}>Ceci est votre page de profil </Text>

   <Button title="Retour" onPress={() => navigation.goBack()} />
  </View>
```

```
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
    backgroundColor: '#fffbe7',
  },
  title: {
    fontSize: 22,
    fontWeight: 'bold',
    marginBottom: 10,
  },
  subtitle: {
    fontSize: 16,
    marginBottom: 20,
  },
});
```

- **route.params** : contient les données envoyées depuis l'écran précédent.
- **navigation.goBack()** : revient à l'écran précédent de la pile.

5. Personnalisation du Header

Vous pouvez personnaliser le header d'un écran, par exemple dans App.js :

```
<Stack.Screen
  name="Profil"
  component={Profil}
  options={{
    title: 'Mon Profil ',
    headerStyle: { backgroundColor: '#007AFF' },
    headerTintColor: '#fff',
  }}
/>
```

Exercice d'application

Consigne

Créer une application avec **3 écrans** :

1. **Accueil** : bouton → aller à **Détails**
2. **Détails** : affiche un texte et bouton → aller à **Profil**
3. **Profil** : récupère un paramètre `nom` envoyé depuis Détails

Supplément

- Personnaliser la couleur du header
- Ajouter un bouton « Retour à l'accueil » dans Profil