Marquer comme terminé

Variables de contexte

Les contextes seront définis dans un fichier spécifique: `Contexte/Context.js`. On pourra définir un contexte particulier pour chaque variable à gérer\ :

```
import React from 'react';

export const TokenContext = React.createContext();

export const UsernameContext = React.createContext();
```

Provider

Voici le composant principal dans App.js, pourvu d'un contexte et son Provider\ :

```
import { TokenContext, UsernameContext } from './Context/Context'

export default function App () {
  const [token, setToken] = useState(null)
  const [username, setUsername] = useState(null)

  console.log('token', token)
  return (
    <UsernameContext.Provider value={[username, setUsername]}>
      <TokenContext.Provider value={[token, setToken]}>
        <Navigation />
      </TokenContext.Provider>
    </UsernameContext.Provider>
  )
}
```

Consumer

Dans les composants fonction, on pourra préférer utiliser le hook useContext. Cela permet d'alléger le code\ :

```
export default function HomeScreen () {
  const [username, setUsername] = useContext(UsernameContext)
  return (
    <>
      <Text>Welcome !</Text>
      <Text>You are logged as {username}</Text>
    </>
  )
}
```

Empilement de l'écran de détails

```
...
import { createNativeStackNavigator } from '@react-navigation/native-stack';
...
const Stack = createNativeStackNavigator()
...

function NavigationTodo () {
  return (
      <Stack.Navigator initialRouteName='List'>
        <Stack.Screen name='List' component={TodoListsScreen} />
        <Stack.Screen name='Details' component={TodoListDetailsScreen} />
      </Stack.Navigator>
  )
}
```

`TodoListsScreen` donne la liste des `TodoList` `TodoListDetailsScreen` donne la liste des `TodoItem` (objet Todo dans l'API Graphql) d'une `Todolist`.

Le composant `TodoLists` liste les `TodoList`. Noter que la navigation est passée au composant `TodoListItem` qui effectue le rendu d'un élément de la liste des `TodoList`.

```
export default function TodoLists (props) {
  return (
    <FlatList
      data={props.data}
      renderItem={({ item }) => <TodoListItem item={item} delete={props.delete} navigation={props.navigation} />}
    />
  )
}
```

`TodoListItem`, représente une `TodoList` dans la liste des `TodoList`:

```
export default function TodoListItem (props) {
  return (
    <TouchableOpacity onPress={() => props.navigation.navigate('Details', {id: props.item.id})}>
      <Item
        id={props.item.id}
        title={props.item.title}
        delete={() => props.delete(props.item.id)}
      />
    </TouchableOpacity>
  )
}
```

L'écran `Details` appelle un composant `TodoListDetails` capte l'identifiant de la `TodoList` :

```
export default function TodoListDetails ({ navigation, route }) {

useEffect(() => {
      getTodos(route.params.id, token)
        .then(
        )
    }
  }, [data])
```

Navigation sélective (en fonction de la présence du jeton)

```
...
import { createBottomTabNavigator } from '@react-navigation/bottom-tabs'

const Tab = createBottomTabNavigator()
...

export default function Navigation () {
  return (
    <TokenContext.Consumer>
      {([token, setToken]) => (
        <NavigationContainer>
          {token == null ? (
            <Tab.Navigator>
              <Tab.Screen name='SignIn' component={SignInScreen} />
              <Tab.Screen name='SignUp' component={SignUpScreen} />
            </Tab.Navigator>
          ) : (
            <Tab.Navigator>
              <Tab.Screen name='Home' component={HomeScreen} />
              <Tab.Screen name='TodoLists' component={NavigationTodo} />
              <Tab.Screen name='SignOut' component={SignOutScreen} />
            </Tab.Navigator>
          )}
        </NavigationContainer>
      )}
    </TokenContext.Consumer>
  )
}
```

ou avec `useContext()` :

?

```
export default function Navigation () {
  const [token, setToken] = useContext(TokenContext)
  return (
        <NavigationContainer>
          {token == null ? (
            <Tab.Navigator>
              <Tab.Screen name='SignIn' component={SignInScreen} />
              <Tab.Screen name='SignUp' component={SignUpScreen} />
            </Tab.Navigator>
          ) : (
            <Tab.Navigator>
              <Tab.Screen name='Home' component={HomeScreen} />
              <Tab.Screen name='TodoLists' component={TodoListsScreen} />
              <Tab.Screen name='SignOut' component={SignOutScreen} />
            </Tab.Navigator>
          )}
        </NavigationContainer>
  )
}
```

Utilisation de l'API dans le code React

On est dans un composant avec deux textInput (username, password) et à la soumission du formulaire, on fait :

```
const onSubmit = () => {
signIn(login, password)
      .then(token => {
        setToken(token)
        setUsername(login)
        props.navigate('Home')
      })
      .catch(err => {
        setError(err.message)
      })
}
```

Modifié le: jeudi 10 octobre 2024, 16:17

◄ TP 4

Choisir un élément

Aller à...

CM 5 ►

?