# Lab 1 : Composants de Base REAN

# 1) Considérer le code ci-dessus (formulaire de création de compte), y ajouter le logo de plb tout en haut, un style avec la méthode createStyle pour chaque composant native(Text, TextInput)

# 2) Ajouter d’autres champs tel que votre email et telephone.

# 3) Le champs password doit être caché lors de la saisie.

# 4)Les boutons Radio et les CheckBox

# Radio : O Etudiant ou O Pro

# CheckBox : Vous aimez : [] Ionic [] React Native

# 

# Une fois vous cliquer sur inscription, on bascule vers la nouvelle interface qui affiche le message, « Votre compte a été créer avec succès », voir l’exemple avec Stack Navigator

# Commencer par ajouter les dépendances suivantes :

# npx expo install react-native-web@~0.18.7 react-dom@18.0.0 @expo/webpack-config@^0.17.0

# Les dépendances suivantes pour les boutons Radio et les CheckBox :

# npm i react-native-paper

import React, { useState } from "react";

import { ScrollView, StyleSheet, View, Text, Image, TextInput, Button } from "react-native";

import { Checkbox, RadioButton } from "react-native-paper";

const App = () => {

  const [name, onChangeName] = useState("");

  const [email, onChangeEmail] = useState("");

  const [password, onChangePassword] = useState("");

  const [value, setValue] = useState("first");

  const [checkedReact, setCheckedReact] = useState(false);

  const [checkedReactNative, setCheckedReactNative] = useState(false);

  const [isSelected, setSelection] = useState(false);

  return (

    <ScrollView style={{ padding: 10 }}>

      <Image

        style={styles.logo}

        source={{

          uri: 'https://pbs.twimg.com/profile\_images/1337422975151255553/AkeDXoIV\_400x400.png',

        }} />

      <Text>Formulaire d'inscription:</Text>

      <View style={{ marginTop: 20, marginBottom: 20, flex: 1, flexDirection: 'row', alignItems: 'center' }}>

        <Text style={styles.label}>Nom: </Text>

        <TextInput

          style={styles.input}

          onChangeText={onChangeName}

          value={name}

          placeholder="Name"

        />

      </View>

      <View style={{ marginTop: 20, marginBottom: 20, flex: 1, flexDirection: 'row', alignItems: 'center' }}>

        <Text style={styles.label}>Email: </Text>

        <TextInput

          style={styles.input}

          onChangeText={onChangeEmail}

          value={email}

          placeholder="Email"

        />

      </View>

      <View style={{ marginTop: 20, marginBottom: 20, flex: 1, flexDirection: 'row', alignItems: 'center' }}>

        <Text style={styles.label}>Password: </Text>

        <TextInput

          style={styles.input}

          onChangeText={onChangePassword}

          value={password}

          placeholder="Password"

          keyboardType="numeric"

          textContentType={password}

          secureTextEntry={true}

        />

      </View>

      <Text>Vous êtes :</Text>

      <RadioButton.Group onValueChange={newValue => setValue(newValue)} value={value}>

        <View style={styles.radio}>

          <RadioButton value="first" />

          <Text>Pro</Text>

        </View>

        <View style={styles.radio}>

          <RadioButton value="second" />

          <Text>Etudiant</Text>

        </View>

      </RadioButton.Group>

      <Text>Vous préférer:</Text>

      <View style={styles.checkbox}>

        <Checkbox

          status={checkedReact ? 'checked' : 'unchecked'}

          onPress={() => {

            setCheckedReact(!checkedReact);

          }}

        />

        <Text>React</Text>

      </View>

      <View style={styles.checkbox}>

        <Checkbox

          status={checkedReactNative ? 'checked' : 'unchecked'}

          onPress={() => {

            setCheckedReactNative(!checkedReactNative);

          }}

        />

        <Text>React Native</Text>

      </View>

      <View style={styles.button}>

        <Button title="Register" />

      </View>

    </ScrollView>

  );

};

const styles = StyleSheet.create({

  logo: {

    height: 50,

    width: 50,

    margin: 'auto'

  },

  label: {

    flex: 0.1

  },

  input: {

    flex: 0.9,

    height: 40,

    margin: 12,

    borderWidth: 1,

    padding: 10,

  },

  button: {

    marginTop: 20

  },

  checkbox: {

    flexDirection: "row",

    alignItems: "center"

  },

  radio: {

    flexDirection: "row",

    alignItems: "center"

  },

  container: {

    flex: 1,

    alignItems: "center",

    justifyContent: "center",

  }

});

export default App;

# Exemple Flatlist

import React from 'react';

import { SafeAreaView, View, FlatList, StyleSheet, Text, StatusBar } from 'react-native';

const DATA = [

  {

    id: 'bd7acbea-c1b1-46c2-aed5-3ad53abb28ba',

    title: 'First Item',

  },

  {

    id: '3ac68afc-c605-48d3-a4f8-fbd91aa97f63',

    title: 'Second Item',

  },

  {

    id: '58694a0f-3da1-471f-bd96-145571e29d72',

    title: 'Third Item',

  },

];

const Item = ({ title }) => (

  <View style={styles.item}>

    <Text style={styles.title}>{title}</Text>

  </View>

);

const App = () => {

  const renderItem = ({ item }) => (

    <Item title={item.title} />

  );

  return (

    <SafeAreaView style={styles.container}>

      <FlatList

        data={DATA}

        renderItem={renderItem}

        keyExtractor={item => item.id}

      />

    </SafeAreaView>

  );

}

const styles = StyleSheet.create({

  container: {

    flex: 1,

    marginTop: StatusBar.currentHeight || 0,

  },

  item: {

    backgroundColor: '#f9c2ff',

    padding: 20,

    marginVertical: 8,

    marginHorizontal: 16,

  },

  title: {

    fontSize: 32,

  },

});

export default App;

# Exemple Flatlist-Selectable

import React, { useState } from "react";

import { FlatList, SafeAreaView, StatusBar, StyleSheet, Text, TouchableOpacity } from "react-native";

const DATA = [

  {

    id: "bd7acbea-c1b1-46c2-aed5-3ad53abb28ba",

    title: "First Item",

  },

  {

    id: "3ac68afc-c605-48d3-a4f8-fbd91aa97f63",

    title: "Second Item",

  },

  {

    id: "58694a0f-3da1-471f-bd96-145571e29d72",

    title: "Third Item",

  },

];

const Item = ({ item, onPress, backgroundColor, textColor }) => (

  <TouchableOpacity onPress={onPress} style={[styles.item, backgroundColor]}>

    <Text style={[styles.title, textColor]}>{item.title}</Text>

  </TouchableOpacity>

);

const App = () => {

  const [selectedId, setSelectedId] = useState(null);

  const renderItem = ({ item }) => {

    const backgroundColor = item.id === selectedId ? "#6e3b6e" : "#f9c2ff";

    const color = item.id === selectedId ? 'white' : 'black';

    return (

      <Item

        item={item}

        onPress={() => setSelectedId(item.id)}

        backgroundColor={{ backgroundColor }}

        textColor={{ color }}

      />

    );

  };

  return (

    <SafeAreaView style={styles.container}>

      <FlatList

        data={DATA}

        renderItem={renderItem}

        keyExtractor={(item) => item.id}

        extraData={selectedId}

      />

    </SafeAreaView>

  );

};

const styles = StyleSheet.create({

  container: {

    flex: 1,

    marginTop: StatusBar.currentHeight || 0,

  },

  item: {

    padding: 20,

    marginVertical: 8,

    marginHorizontal: 16,

  },

  title: {

    fontSize: 32,

  },

});

export default App;

# Exercice 15 min

# 

/\*

import \* as React from 'react';

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

import { NavigationContainer } from '@react-navigation/native';

import { createNativeStackNavigator } from '@react-navigation/native-stack';

const Stack = createNativeStackNavigator();

const HomeScreen = ({ navigation }) => {

  return (

    <Button

      title="Go to Jane's profile"

      onPress={() =>

        navigation.navigate('Profile', { name: 'Jane' , adresse:'Paris'})

      }

    />

  );

};

const ProfileScreen = ({ navigation, route }) => {

  return (

  <View>

  <Text>This is {route.params.name}'s profile</Text>

  <Text>This is {route.params. adresse}</Text>

  </View>)

  ;

};

const App = () => {

  return (

    <NavigationContainer>

    <Stack.Navigator>

      <Stack.Screen name="Home" component={HomeScreen} options={{ title: 'Welcome' }} />

      <Stack.Screen name="Profile" component={ProfileScreen} options={{ title: 'Me' }} />

    </Stack.Navigator>

  </NavigationContainer>

  );

};

export default App;

\*/

/\*

import React from 'react';

import { SafeAreaView, View, FlatList, StyleSheet, Text, StatusBar } from 'react-native';

const DATA = [

  {

    id: 'bd7acbea-c1b1-46c2-aed5-3ad53abb28ba',

    title: 'First Item',

  },

  {

    id: '3ac68afc-c605-48d3-a4f8-fbd91aa97f63',

    title: 'Second Item',

  },

  {

    id: '58694a0f-3da1-471f-bd96-145571e29d72',

    title: 'Third Item',

  },

  {

    id: 'bd7acbea-c1b1-46c2-aed5-3ad53abb28ba',

    title: 'First Item',

  },

  {

    id: '3ac68afc-c605-48d3-a4f8-fbd91aa97f63',

    title: 'Second Item',

  },

  {

    id: '58694a0f-3da1-471f-bd96-145571e29d72',

    title: 'Third Item',

  },

  {

    id: 'bd7acbea-c1b1-46c2-aed5-3ad53abb28ba',

    title: 'First Item',

  },

  {

    id: '3ac68afc-c605-48d3-a4f8-fbd91aa97f63',

    title: 'Second Item',

  },

  {

    id: '58694a0f-3da1-471f-bd96-145571e29d72',

    title: 'Third Item',

  },

  {

    id: 'bd7acbea-c1b1-46c2-aed5-3ad53abb28ba',

    title: 'First Item',

  },

  {

    id: '3ac68afc-c605-48d3-a4f8-fbd91aa97f63',

    title: 'Second Item',

  },

  {

    id: '58694a0f-3da1-471f-bd96-145571e29d72',

    title: 'Third Item',

  },

];

const Item = ({ title }) => (

  <View style={styles.item}>

    <Text style={styles.title}>{title}</Text>

  </View>

);

const App = () => {

  const renderItem = ({ item }) => (

    <Item title={item.title} />

  );

  return (

    <SafeAreaView style={styles.container}>

      <FlatList

        data={DATA}

        renderItem={renderItem}

        keyExtractor={item => item.id}

      />

    </SafeAreaView>

  );

}

const styles = StyleSheet.create({

  container: {

    flex: 1,

    marginTop: StatusBar.currentHeight || 0,

  },

  item: {

    backgroundColor: '#f9c2ff',

    padding: 20,

    marginVertical: 8,

    marginHorizontal: 16,

  },

  title: {

    fontSize: 32,

  },

});

export default App;\*/

import React, { useState } from "react";

import {View,Image, FlatList, SafeAreaView, StatusBar, StyleSheet, Text, TouchableOpacity } from "react-native";

  /\*{

    id: "bd7acbea-c1b1-46c2-aed5-3ad53abb28ba",

    title: "First Item",

  },

  {

    id: "3ac68afc-c605-48d3-a4f8-fbd91aa97f63",

    title: "Second Item",

  },

  {

    id: "58694a0f-3da1-471f-bd96-145571e29d72",

    title: "Third Item",

  },\*/

>

  const DATA = [

    {

      id: "bd7acbea-c1b1-46c2-aed5-3ad53abb28ba",

      img: "https://img.cuisineaz.com/660x660/2013/12/20/i18445-margherite.webp",

      prix: "10€",

      ingredients: "Tomate, Fromage",

      title: "Marguerite",

    },

    {

      id: "3ac68afc-c605-48d3-a4f8-fbd91aa97f63",

      img: "https://cdn.pratico-pratiques.com/app/uploads/sites/3/2018/08/15142009/pizza-aux-fruits-de-mer.jpg",

      prix: "11€",

      ingredients: "Tomate, Fromage, fruits de mer",

      title: "Fruit de mer",

    },

    {

      id: "58694a0f-3da1-471f-bd96-145571e29d72",

      img: "https://assets.afcdn.com/recipe/20161130/7916\_w1024h778c1cx2808cy1872.webp",

      prix: "15€",

      ingredients: "Tomate, Fromage, jambon",

      title: "Calzone",

    },

  ];

const Item = ({ item, onPress, backgroundColor, textColor }) => (

  <TouchableOpacity onPress={onPress} style={[styles.item, backgroundColor]}>

    <View style={styles.row}>

        <Image style={styles.logo}  source={{ uri: item.img}}/>

        <View style={styles.col}>

           <Text style={[styles.title, textColor]}>Titre : {item.title}</Text>

           <Text style={[styles.title, textColor]}>Prix : {item.prix}</Text>

           <Text style={[styles.title, textColor]}>Ingrédients : {item.ingredients}</Text>

        </View>

     </View>

  </TouchableOpacity

);

const App = () => {

  const [selectedId, setSelectedId] = useState(null);

  const monItem = ({ item }) => {

    const backgroundColor = item.id === selectedId ? "#6e3b6e" : "#f9c2ff";

    const color = item.id === selectedId ? 'white' : 'black';

    return (

      <Item

        item={item}

        onPress={() => setSelectedId(item.id)}

        backgroundColor={{ backgroundColor }}

        textColor={{ color }}

      />

    );

  };

  return (

    <SafeAreaView style={styles.container}>

      <FlatList

        data={DATA}

        renderItem={monItem}

        keyExtractor={(item) => item.id}

        extraData={selectedId}

      />

    </SafeAreaView>

  );

};

const styles = StyleSheet.create({

  container: {

    flex: 1,

    marginTop: StatusBar.currentHeight || 0,

  },

  item: {

    padding: 20,

    marginVertical: 8,

    marginHorizontal: 16,

  },

  title: {

    fontSize: 18,

  },

  row: {

    flex:1,

    flexDirection: "row",

    flexWrap: "wrap",

  },

  col: {

    flex:0.5,

    flexDirection: "col",

    flexWrap: "wrap",

    marginHorizontal:10

  },

  logo: {

    flex:0.5,

    width: 150,

    height: 150,

  },

});

export default App;

## **Partie 1 : React Navigation**

# <https://reactnative.dev/docs/navigation#react-navigation>

# Exemple 1 :Utilisant StackNavigator

First, you need to install them in your project:

npm install @react-navigation/native @react-navigation/native-stack

Next, install the required peer dependencies. You need to run different commands depending on whether your project is an Expo managed project or a bare React Native project.

* If you have an Expo managed project, install the dependencies with expo:

expo install react-native-screens react-native-safe-area-context

* If you have a bare React Native project, install the dependencies with npm:

npm install react-native-screens react-native-safe-area-context

"dependencies": {

    "@expo/webpack-config": "^0.17.0",

    "@react-navigation/native": "^6.0.12",

    "@react-navigation/native-stack": "^6.8.0",

    "expo": "~46.0.9",

    "expo-status-bar": "~1.4.0",

    "react": "18.0.0",

    "react-dom": "18.0.0",

    "react-native": "0.69.5",

    "react-native-paper": "^4.12.4",

    "react-native-web": "~0.18.7",

    "react-native-screens": "~3.15.0",

    "react-native-safe-area-context": "4.3.1"

  },

# Now, you need to wrap the whole app in NavigationContainer. Usually you'd do this in your entry file, such as index.js or App.js:

### Usage[​](https://reactnative.dev/docs/navigation#usage)

Now you can create an app with a home screen and a profile screen:

In this example, there are 2 screens (Home and Profile) defined using the Stack.Screen component. Similarly, you can define as many screens as you like.

You can set options such as the screen title for each screen in the options prop of Stack.Screen.

Each screen takes a component prop that is a React component. Those components receive a prop called navigation which has various methods to link to other screens. For example, you can use navigation.navigate to go to the Profile screen:

import \* as React from 'react';

import {Button,Text } from 'react-native';

import { NavigationContainer } from '@react-navigation/native';

import { createNativeStackNavigator } from '@react-navigation/native-stack';

const Stack = createNativeStackNavigator();

const HomeScreen = ({ navigation }) => {

  return (

    <Button

      title="Go to Jane's profile"

      onPress={() =>

        navigation.navigate('Profile', { name: 'Jane' })

      }

    />

  );

};

const ProfileScreen = ({ navigation, route }) => {

  return <Text>This is {route.params.name}'s profile</Text>;

};

const App = () => {

  return (

    <NavigationContainer>

    <Stack.Navigator>

      <Stack.Screen

        name="Home"

        component={HomeScreen}

        options={{ title: 'Welcome' }}

      />

      <Stack.Screen name="Profile" component={ProfileScreen} options={{ title: 'Me' }} />

    </Stack.Navigator>

  </NavigationContainer>

  );

};

export default App;

# Exemple 2 : Utilisant TabNavigator avec Icons

npm install react-navigation

npm install @react-navigation/native

npm install @react-navigation/native-stack

npm install react-native-safe-area-context

npm install @react-navigation/bottom-tabs

Attention : Les noms des composants doivent toujours commencer par une majuscule !

import { StatusBar } from 'expo-status-bar';

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

import { NavigationContainer } from '@react-navigation/native';

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

const tab = createBottomTabNavigator();

function AccueilScreen() {

  return (<View style={styles.container}>

    <Text>Welcome to REACT native</Text>

    <StatusBar style="auto" />

  </View>);

}

function ContactScreen() {

  return (<View style={styles.container}>

    <Text>Leave a message here</Text>

    <StatusBar style="auto" />

  </View>);

}

export default function App() {

  return (

    <NavigationContainer>

      <tab.Navigator>

       <tab.Screen name="Home" component={AccueilScreen} options={{ title: 'Accueil' }}/>

       <tab.Screen name="Contact" component={ContactScreen} options={{ title: 'Call us' }}/>

      </tab.Navigator>

    </NavigationContainer>

  );

}

const styles = StyleSheet.create({

  container: {

    flex: 1,

    backgroundColor: '#fff',

    alignItems: 'center',

    justifyContent: 'center',

  },

});

Ajoutons maintenant des icones de navigation 😊

<https://ionic.io/ionicons>

<https://reactnavigation.org/docs/screen-options/>

Une image contenant texte

Description générée automatiquement

import { StatusBar } from 'expo-status-bar';

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

import { NavigationContainer } from '@react-navigation/native';

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

import Ionicons from 'react-native-vector-icons/Ionicons';

const tab = createBottomTabNavigator();

function AccueilScreen() {

  return (<View style={styles.container}>

    <Text>Welcome to REACT native</Text>

    <StatusBar style="auto" />

  </View>);

}

function SettingScreen() {

  return (<View style={styles.container}>

    <Text>Vos paramètres</Text>

    <StatusBar style="auto" />

  </View>);

}

export default function App() {

  return (

    <NavigationContainer>

      <tab.Navigator

        screenOptions={({ route }) => ({

          tabBarIcon: ({ focused, color, size }) => {

            let iconName;

            if (route.name == "Home") { iconName = "home-outline"; }

            else if (route.name == "Setting") { iconName = "settings-outline"; }

            return (

              <Ionicons

                name={iconName}

                color={color}

                size={size}

              />

            );

          },

        })}

      >

        <tab.Screen name="Home" component={AccueilScreen} />

        <tab.Screen name="Setting" component={SettingScreen} />

      </tab.Navigator>

    </NavigationContainer>

  );

}

const styles = StyleSheet.create({

  container: {

    flex: 1,

    backgroundColor: '#fff',

    alignItems: 'center',

    justifyContent: 'center',

  },

});

On peut agir sur les autres options de taille et couleur :

<Ionicons

                name={iconName}

                color={'red'}

                size={20}

              />

Exercice

1. Créer une collection de 3 utilisateurs (id, nom, email, adresse, photo, tel, age…)
2. Utiliser un FlatList Selectable pour afficher cette collection(Utiliser un Item)
3. Développer un autre screen permettant d’afficher plus d’infos sur l’utilisateur si vous cliquer là-dessus.

Astuce : onPress={() =>

        navigation.navigate('Profile', { name: 'Jane' })}

import React, { useState } from "react";  
import { SafeAreaView, Button, Text, View, Image, FlatList, StyleSheet, StatusBar, TouchableOpacity } from 'react-native';  
import { NavigationContainer } from '@react-navigation/native';  
import { createNativeStackNavigator } from '@react-navigation/native-stack';  
import { createBottomTabNavigator } from '@react-navigation/bottom-tabs';

const Stack = createNativeStackNavigator();  
const Tab = createBottomTabNavigator();

const DATA = [

  {  
    id: "bd7acbea-c1b1-46c2-aed5-3ad53abb28ba",  
    firstName: 'Dimitri',  
    lastName: 'CERRATOPS',  
    description: "User 1 : Lorem ipsum",

  },  
  {  
    id: "3ac68afc-c605-48d3-a4f8-fbd91aa97f63",  
    firstName: 'Sarah',  
    lastName: 'FISTOL',  
    description: "User 2 : Lorem ipsum",  
  },  
  {

    id: "58694a0f-3da1-471f-bd96-145571e29d72",  
    firstName: 'Hervé',  
    lastName: 'LOCIRAPTOR',  
    description: "User 3 : Lorem ipsum",  
  },

];

const Item = ({ item, onPress, backgroundColor, textColor }) => (  
<TouchableOpacity onPress={onPress} style={[styles.item, backgroundColor]}>  
<View style={{flexDirection: 'row'}}>  
      {/\*<Image  
          style={styles.picture}  
          source={{ uri: item.img}}  
        />\*/}  
<View>  
<Text style={[styles.title, textColor]}>{item.firstName}</Text>  
<Text style={[styles.title, textColor]}>{item.lastName}</Text>  
</View>  
</View>    
</TouchableOpacity>  
);

const HomeScreen = ({ navigation }) => {  
  return (  
<SafeAreaView style={styles.container}>  
<Button  
        title="Go to Jane's profile"  
        onPress={() =>  
          navigation.navigate('Profile', { name: 'Jane' , adresse:'Paris'})  
        }  
      />  
<Button  
        title="Our Users"  
        onPress={() =>  
          navigation.navigate('User')  
        }  
      />  
</SafeAreaView>  
  );  
};

const ProfileScreen = ({ navigation, route }) => {  
  return (  
<SafeAreaView style={styles.container}>  
<Text>This is {route.params.name}'s profile</Text>  
<Text>This is {route.params.adresse}</Text>  
</SafeAreaView>)  
  ;  
};

const UserScreen = ({navigation}) => {

  const [selectedId, setSelectedId] = useState(null);

  const renderItem = ({ item }) => {

    const backgroundColor = item.id === selectedId ? "#6e3b6e" : "#f9c2ff";  
    const color = item.id === selectedId ? 'white' : 'black';

    return (  
<Item   
        item={item}  
        onPress={() => navigation.navigate('UserDetails', {userId: item.id})}  
        backgroundColor={{ backgroundColor }}  
        textColor={{ color }}  
      />  
    )

  };

  return (  
<SafeAreaView style={styles.container}>  
<Text>Our Users:</Text>  
<FlatList  
        data={DATA}  
        renderItem={renderItem}  
        keyExtractor={item => item.id}  
      />  
</SafeAreaView>  
  );  
};

const UserDetails = ({navigation, route}) => {  
  return (  
<SafeAreaView style={styles.container}>  
<Text>User: {route.params.userId}</Text>  
</SafeAreaView>  
  );  
}

const Nav = () => {  
  return (  
<Tab.Navigator>  
<Tab.Screen name="Home" component={HomeScreen}/>  
<Tab.Screen   
          name="Profile"  
          component={ProfileScreen}   
          initialParams={{ name: 'Jane' , adresse:'Paris'}}  
        />  
<Tab.Screen name="User" component={UserScreen} options={{ title: 'Our Users' }} />  
</Tab.Navigator>  
  )  
}

const App = () => {  
  return (  
<NavigationContainer>  
<StatusBar  
        animated={true}  
        backgroundColor="#f9c2ff" />  
<Stack.Navigator>  
<Stack.Screen name="Nav" component={Nav} options={{ title: 'Home', headerShown: false }}/>  
<Stack.Screen name="UserDetails" component={UserDetails}  options={{ user: 1 }}/>  
</Stack.Navigator>  
</NavigationContainer>  
  );  
};

const styles = StyleSheet.create({  
  container: {  
    flex: 1,  
    marginTop: StatusBar.currentHeight || 0  
  },  
  item: {  
    backgroundColor: '#f9c2ff',  
    padding: 20,  
    marginVertical: 8,  
    marginHorizontal: 16  
  },  
  title: {  
    fontSize: 24,  
  },  
  picture: {  
    height: 100,  
    width: 100,  
    marginRight: 16  
  },  
});

export default App;

## **Partie 2 : Consommation des APIs via le Bundle Fetch**

On s’intéresse dans cette section à la consommation de apis suivantes :

<https://jsonplaceholder.typicode.com/>



# Exemple 1 : Get All Data

import { StatusBar } from 'expo-status-bar';

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

import { NavigationContainer } from '@react-navigation/native';

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

import Ionicons from 'react-native-vector-icons/Ionicons';

import { useEffect, useState } from 'react';

const tab = createBottomTabNavigator();

function AccueilScreen() {

  const[users,setUsers]=useState([]);

  /// Chargement des data

  useEffect(() => {

    getData();

  },[])

  const getData=()=>{

    fetch('https://jsonplaceholder.typicode.com/users')

    .then(response=>response.json())

    .then(data=>{

      //console.log(data);

      setUsers(data);

    })

  }

  return (<View style={styles.container}>

    {console.log(users)}

    <Text>Welcome to REACT native</Text>

    <StatusBar style="auto" />

    <View style={styles.container}>

      {users.map(\_user=><Text key={\_user.id}>{\_user.name}\*\*\*{\_user.email}</Text>)}

    </View>

  </View>);

}

function SettingScreen() {

  return (<View style={styles.container}>

    <Text>Vos paramètres</Text>

    <StatusBar style="auto" />

  </View>);

}

export default function App() {

  return (

    <NavigationContainer>

      <tab.Navigator

        screenOptions={({ route }) => ({

          tabBarIcon: ({ focused, color, size }) => {

            let iconName;

            if (route.name == "Home") { iconName = "home-outline"; }

            else if (route.name == "Setting") { iconName = "settings-outline"; }

            return (

              <Ionicons

                name={iconName}

                color={'red'}

                size={size}

              />

            );

          },

        })}

      >

        <tab.Screen name="Home" component={AccueilScreen} />

        <tab.Screen name="Setting" component={SettingScreen} />

      </tab.Navigator>

    </NavigationContainer>

  );

}

const styles = StyleSheet.create({

  container: {

    flex: 1,

    backgroundColor: '#fff',

    alignItems: 'center',

    justifyContent: 'center',

  },

});

We can use **async/await**, available in Es6, to refactor our code.

const getData=()=>{

    fetch('https://jsonplaceholder.typicode.com/users')

    .then(response=>response.json())

    .then(data=>{

      //console.log(data);

      setUsers(data);

    })

  }

We first make our **getData** function asynchronous by attaching the **async** keyword next to its name. Then, use **await** for any asynchronous line of code inside it. The above code makes the entire fetching data action cleaner with the same result.

  const getData=async()=>{

    const response=await fetch('https://jsonplaceholder.typicode.com/users');

    const data=await response.json();

    setUsers(data)

  }

## Error Handling in Async/Await

When we used the **.then()** blocks, we could use the .**catch** block to handle our errors. However, with async/await we can no longer use the **.then** and **.catch** methods directly.

So, how do we handle errors?

We can use the try/catch blocks in this case. Everything related to processing the data and making the Fetch call can be wrapped inside a try block. Then, we can have a catch block that automatically takes an error object as a parameter.

We can then do anything inside this catch block to handle that error.

Here's what the updated **getData** function would look like with try/catch blocks:

  const getData=async()=>{

    try{

      const response=await fetch('https://jsonplaceholder.typicode.com/users');

      const data=await response.json();

      setUsers(data)

    }

    catch(error){

      console.log(error)

    }

  }

Par exemple avec une faute dans l’url :

https://jsonplaceholder.typixxcode.com/users



## Implement Loading Action With Fetch

We know that getting data using the Fetch API is an asynchronous action. This means that the data is not displayed instantly to the user. Now, this might create a bad user experience for your app because it might leave users wondering what really is going on.

Therefore, in most cases, the data-fetching action is accompanied by a loader. Until you receive and process the response from your Fetch request, you can show a loader to the user to indicate that a data-fetching action is underway.

### Use the **.finally** Handler in Fetch

However, with try/catch blocks on Fetch, we can handle the loading state very smoothly via the**.finally** handler. First, we'll create a new state that would store the loading status:

Then, we'll set this state to **loading**inside our **useEffect** right before we invoke the **getData** function. We'll also intentionally delay the **getData** function's calling using a **setTimeout**.

useEffect(() => {

    //getData();

    setFetchedState('loading')

    setTimeout(()=>getData(),3000);

  },[])

Now, we'll need to set the loading state back to null after the request is completed. Usually, you might do it two times**—**once inside the **try** block then again inside the **catch** block.

That's necessary because you also need to toggle the loading state in case of errors. However, we use the **finally** block to only toggle the loading state once.

const getData=async()=>{

    try{

      const response=await fetch('https://jsonplaceholder.typicode.com/users');

      const data=await response.json();

      setUsers(data)

    }

    catch(error){

      console.log(error)

    }

    finally{

      setFetchedState(null);

    }

  }

The**finally** block is executed after the request is complete regardless of its success or error. Thus, even if your request fails, the **finally** block will run.

Now, let's conditionally render a loading message on the UI:

  return (<View style={styles.container}>

    <Text>Welcome to REACT native</Text>

    <StatusBar style="auto" />

    <View style={styles.container}>

      {

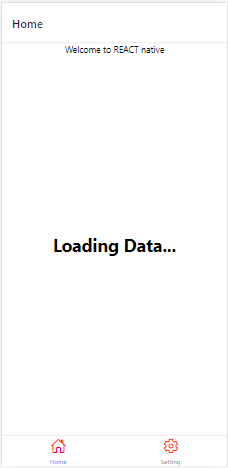
      fetchedState ? <Text style={styles.loadingtext}>Loading Data...</Text> :

      users.map(\_user=><Text key={\_user.id}>{\_user.name}\*\*\*{\_user.email}</Text>)

      }

    </View>

  </View>);



Avec ActivityIndicator

<View style={styles.container}>

      {

      fetchedState ? <ActivityIndicator size="large" color="#0000ff" />:

      users.map(\_user=><Text key={\_user.id}>{\_user.name}\*\*\*{\_user.email}</Text>)

      }

    </View>



Pour les styles :

const styles = StyleSheet.create({

  container: {

    flex: 1,

    backgroundColor: '#fff',

    alignItems: 'center',

    justifyContent: 'center',

  },

  loadingtext:{

    fontSize: 28,

    fontWeight: 'bold',

  },

  text:{

    fontSize:24,

    margin:10

  }

});

Application : Utiliser un composant FlatList pour afficher les data des users

# Exercice 2 : Get By Id & Navigation & Flast List Selectable

# Exemple d’API : <https://jsonplaceholder.typicode.com/users/1>

# 

# Exercice 3 : Post Data

createUser(user) {

// Simple POST request with a JSON body using fetch

const requestOptions = {

method: 'POST',

headers: { 'Content-Type': 'application/json' },

body: JSON.stringify(user)

};

fetch('https://reqres.in/api/posts', requestOptions)

.then(response => response.json())

.then(data => this.setState({ postId: data.id }));

}

# Solution

const createUser = () => {

    let user = {

      nom:"Amine",

      prenom:"Mez",

      email:"am@gmail.com",

      password:"1234"

    };

    // Simple POST request with a JSON body using fetch

    const requestOptions = {

      method: 'POST',

      headers: { 'Content-Type': 'application/json' },

      body: JSON.stringify(user)

    };

    fetch('https://pharma.tunitransport.com/api/public/index.php/api/pharma/users', requestOptions)

      .then(response => response.json())

      .then(data =>console.log(data));

  }

 <Button title="Ajouter" onPress={createUser}></Button>

import { StatusBar } from 'expo-status-bar';

import { StyleSheet, Text, View, FlatList, SafeAreaView, TouchableOpacity, Image, ActivityIndicator, Button } from 'react-native';

import { NavigationContainer } from '@react-navigation/native';

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

import Ionicons from 'react-native-vector-icons/Ionicons';

import { useEffect, useState } from 'react';

const tab = createBottomTabNavigator();

const Item = ({ item, onPress, backgroundColor, textColor }) => (

  <TouchableOpacity onPress={onPress} style={{ backgroundColor }}>

    <View style={{ flexDirection: 'row' }}>

      <Image

        style={styles.picture}

        source={{ uri: 'https://png.pngtree.com/png-vector/20190704/ourlarge/pngtree-businessman-user-avatar-free-vector-png-image\_1538405.jpg' }}

      />

      <View>

        <Text style={{ textColor }}>{item.name}</Text>

        <Text>{item.email}</Text>

        <Text>{item.nom}</Text>

        <Text>{item.prenom}</Text>

        <Text>{item.password}</Text>

      </View>

    </View>

  </TouchableOpacity>

);

function AccueilScreen() {

  const [users, setUsers] = useState([]);

  const [fetchedState, setFetchedState] = useState(null);

  /// Chargement des data

  /\* useEffect(() => {

     getData();

   }, [])\*/

  useEffect(() => {

    setFetchedState('loading')

    setTimeout(() => getData(), 2000);

  }, [])

  /\*const getData = () => {

    fetch('https://jsonplaceholder.typicode.com/users')

      .then(response => response.json())

      .then(data => {

        //console.log(data);

        setUsers(data);

      })

  }\*/

  const getData = async () => {

    //const response=await fetch('https://jsonplaceholder.typicode.com/users');

    //const data=await response.json();

    //setUsers(data)

    try {

      //const response=await fetch('https://jsonplaceholder.typicode.com/users');

      const response = await fetch('https://pharma.tunitransport.com/api/public/index.php/api/pharma/users');

      const data = await response.json();

      //console.log(data['hydra:member'])

      setUsers(data['hydra:member'])

    }

    catch (error) {

      //console.log(error)

      console.log("Vérifier votre api...");

    }

    finally {

      setFetchedState(null);

    }

  }

  const renderItem = ({ item }) => {

    //const backgroundColor = item.id === selectedId ? "#6e3b6e" : "#f9c2ff";

    //const color = item.id === selectedId ? 'white' : 'black';

    return (

      <Item

        item={item}

        // onPress={() => navigation.navigate('UserDetails', {userId: item.id})}

        backgroundColor='#f9c2ff'

        textColor='blue'

      />

    )

  };

  const createUser = () => {

    let user = {

      nom:"Amine",

      prenom:"Mez",

      email:"am@gmail.com",

      password:"1234"

    };

    // Simple POST request with a JSON body using fetch

    const requestOptions = {

      method: 'POST',

      headers: { 'Content-Type': 'application/json' },

      body: JSON.stringify(user)

    };

    fetch('https://pharma.tunitransport.com/api/public/index.php/api/pharma/users', requestOptions)

      .then(response => response.json())

      .then(data =>console.log(data));

  }

  return (

    <SafeAreaView style={styles.container}>

      <Button title="Ajouter" onPress={createUser}></Button>

      <Text>Our Users:</Text>

      {

        fetchedState ? <ActivityIndicator size="large" color="#0000ff" /> :

          <FlatList

            data={users}

            renderItem={renderItem}

            keyExtractor={item => item.id}

          />

      }

    </SafeAreaView>);

}

function SettingScreen() {

  return (<View style={styles.container}>

    <Text>Vos paramètres</Text>

    <StatusBar style="auto" />

  </View>);

}

export default function App() {

  return (

    <NavigationContainer>

      <tab.Navigator

        screenOptions={({ route }) => ({

          tabBarIcon: ({ focused, color, size }) => {

            let iconName;

            if (route.name == "Home") { iconName = "home-outline"; }

            else if (route.name == "Setting") { iconName = "settings-outline"; }

            return (

              <Ionicons

                name={iconName}

                color={'red'}

                size={size}

              />

            );

          },

        })}

      >

        <tab.Screen name="Home" component={AccueilScreen} />

        <tab.Screen name="Setting" component={SettingScreen} />

      </tab.Navigator>

    </NavigationContainer>

  );

}

const styles = StyleSheet.create({

  container: {

    flex: 1,

    marginTop: StatusBar.currentHeight || 0

  },

  item: {

    backgroundColor: '#f9c2ff',

    padding: 20,

    marginVertical: 8,

    marginHorizontal: 16

  },

  title: {

    fontSize: 24,

  },

  picture: {

    height: 100,

    width: 100,

    marginRight: 16

  },

});

# Exercice : 20 min

# 1)Créer depuis le TabNavigation un autre Screen AddUserScreen

# 2)Développer un formulaire permettant de saisir les infos d’un user : nom, prenom, email et password et un bouton « S’enregister »

# 3)Une fois , enregister l’application doit nous rediriger vers le Screen ListUserd

# Exercice 4 : Delete Data

deleteUser(id) {

// Simple DELETE request with fetch

fetch('https://jsonplaceholder.typicode.com/posts/'+id, { method: 'DELETE' })

.then(() => this.setState({ status: 'Delete successful' }));

}

# Exercice 5 : Update Data

updateUser(userUpdated) {

// Simple PUT request with a JSON body using fetch

const requestOptions = {

method: 'PUT',

headers: { 'Content-Type': 'application/json' },

body: JSON.stringify(userUpdated)

};

fetch('https://jsonplaceholder.typicode.com/posts/1', requestOptions)

.then(response => response.json())

.then(data => this.setState({ postId: data.id }));

}

# Lab 2 : Consommation API & Tab Navigation