

Présentation de la correction de l'exercice

# Objectif de l'exercice

- A partir de l'exemple ci-contre, l'objectif est de récupérer des données JSON et d'afficher les noms, les prénoms et les CP des clients et de changer le style de chacun.
- Pour cela, nous avons à notre disposition un script PHP (GetClientsFromMySQL.php) qui renvoie au format JSON des données d'une base de données.
- Le script PHP est stocké à l'URL suivant :  
`http://pascalbuguet.alwaysdata.net/reactPHP/php/`



# Explication du code : Récupérer les données

```
import React, { useEffect, useState } from 'react';
import { ActivityIndicator, FlatList, Text, View, StyleSheet } from 'react-native';

const App = () => {
  const [isLoading, setLoading] = useState(true);
  const [data, setData] = useState([]);
  useEffect(() => {
    const url = "http://pascalbuguet.alwaysdata.net/reactPHP/php/GetClientsFromMySQL.php";
    fetch(url)
      .then((response) => response.json())
      .then((json) => setData(json))
      .catch((error) => console.error(error))
      .finally(() => setLoading(false));
  }, []);
```

On importe **useEffect** (pour gérer les side effects) et **useState** (pour gérer l'état) de **react**.

On importe **ActivityIndicator**, **FlatList**, **Text**, **View** et **StyleSheet** de **react-native**.

On crée la fonction **App**, on fixe le chargement à **true** et on définit la gestion de l'état data.

Dans **useEffect**, on déclare la constante **url** qui contient le chemin d'accès et le script PHP qui fournit les résultats de l'exécution d'une requête SQL.

Ensuite, on convertit la réponse reçue en données JSON, puis on traite les données JSON.

S'il y a une erreur, on l'affiche dans la console.

On fixe le chargement à **false** et on affiche la liste.

# Explication du code : Afficher les données

```
return (  
  <View style={{ flex: 1, padding: 24 }}>  
    {isLoading ? <ActivityIndicator /> : (  
      <FlatList  
        data={data}  
        keyExtractor={({ id }, index) => id}  
        renderItem={({ item }) => (  
          <View style={styles.view}>  
            <Text style={styles.nom}>{item.nom}</Text>  
            <Text style={styles.prenom}>{item.prenom}</Text>  
            <Text style={styles.cp}>{item.cp}</Text>  
          </View>  
        )  
      )  
    )  
  </View>  
);
```

Dans le **return** de la fonction **App**, on a :  
Une condition, si **isLoading** = True, alors on indique le chargement.  
Sinon on affiche la **FlatList**.

Dans la **FlatList**, on affecte des données à l'attribut **data**.

On extrait une **idKey** pour chaque objet de donnée.

Avec **renderItem**, on affiche les items dans la **FlatList** par rapport à leur key.  
Les items affichés sont le *nom*, le *prénom* et le *cp*.  
Ils sont affichés dans un conteneur **Text** pour chacun.

On affecte un style pour chaque **Text** et pour le **View**.

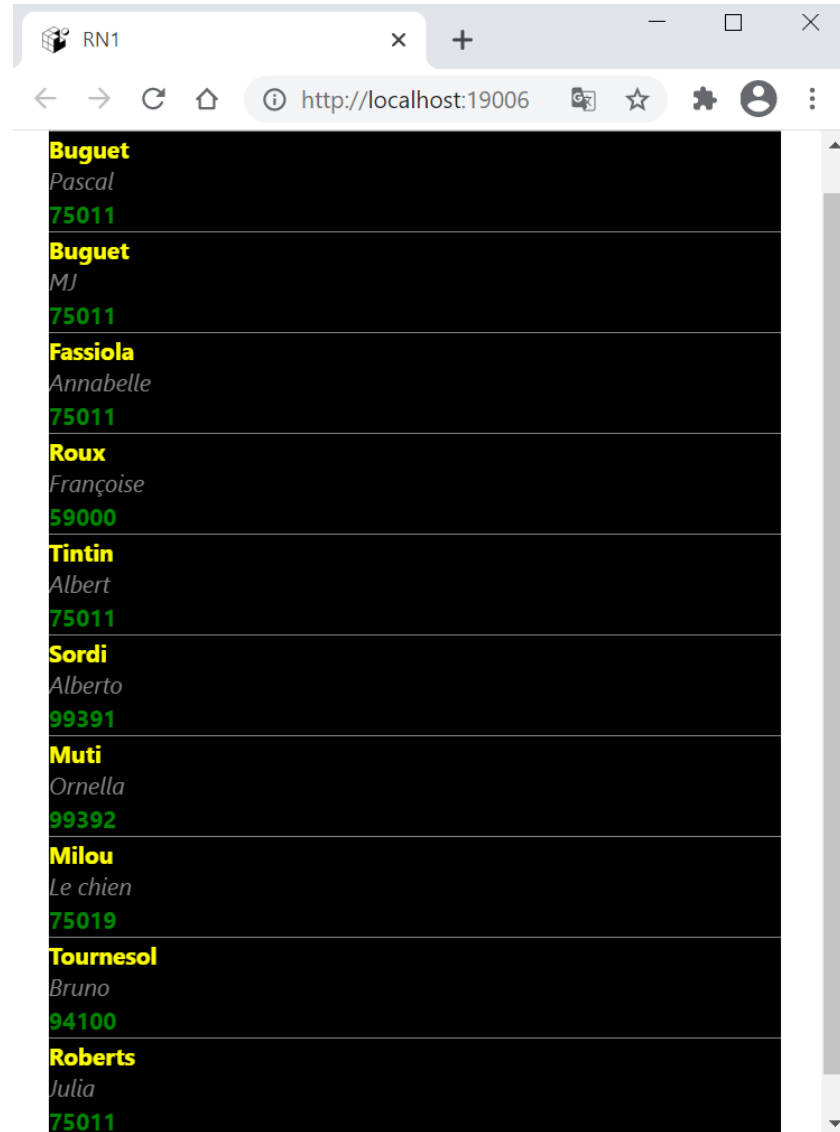
# Explication du code : Ajout des styles

```
const styles = StyleSheet.create({
  nom: {
    height: 20,
    color: 'yellow',
    fontWeight: 'bold',
    borderTopWidth: 1,
    borderTopColor: 'gray'
  },
  prenom: {
    height: 20,
    color: 'gray',
    fontStyle: 'italic',
    borderTopColor: 'gray'
  },
  cp: {
    height: 20,
    color: 'green',
    fontWeight: 'bold',
    borderTopColor: 'gray'
  },
  view: {
    backgroundColor: 'black'
  }
});

export default App;
```

On définit les styles **nom**, **prenom**, **cp** et **view** dans la constante **styles** avec `StyleSheet.create`.

Résultat dans un navigateur :



The screenshot shows a web browser window with the address bar displaying 'http://localhost:19006'. The browser has a single tab titled 'RN1'. The main content area displays a list of names and IDs on a black background. Each entry consists of a name in yellow, a first name in light gray, and an ID in green. The entries are separated by thin white horizontal lines. A vertical scrollbar is visible on the right side of the list.

<b>Buguet</b>
<i>Pascal</i>
<b>75011</b>
<b>Buguet</b>
<i>MJ</i>
<b>75011</b>
<b>Fassiola</b>
<i>Annabelle</i>
<b>75011</b>
<b>Roux</b>
<i>Françoise</i>
<b>59000</b>
<b>Tintin</b>
<i>Albert</i>
<b>75011</b>
<b>Sordi</b>
<i>Alberto</i>
<b>99391</b>
<b>Muti</b>
<i>Ornella</i>
<b>99392</b>
<b>Milou</b>
<i>Le chien</i>
<b>75019</b>
<b>Tournesol</b>
<i>Bruno</i>
<b>94100</b>
<b>Roberts</b>
<i>Julia</i>
<b>75011</b>