# React Native

[Contributeur : Robin GARBACIAK SLAM20-22]

## 1.Installation de React Native

**Par quoi commencer ?**

Pour commencer à développer avec React Native vous avez besoin de Node.js et d'un éditeur de code. Allez dans l'invite de commande et taper :

```
npm install -g expo-cli
```

Sur Mac, pour installer une librairie de manière globale, il faut l'installer en tant qu'administrateur. Vous devrez donc ajouter sudo avant la commande :

```
sudo npm install -g expo-cli
```

Maintenant que vous avez fait ceci, vous avez installé Expo.Sa va vous permettre de lancer l'application que vous allez créer sur votre téléphone.Pour lancer votre application vous avez besoin de Expo Go qui est téléchargeable sur le Google Play et l'App Store.

Pour créer votre premier projet, vous avez besoin de vous mettre dans un dossier de préférence vierge que vous allez créer sur votre ordinateur, appelé le React. Une fois créé allez y via l'invite de commande, dans mon cas le chemin sera:

```
cd D:\React
```

Puis créer votre application via cette commande :

```
expo init NomDuPojet
```

Expo va vous demandez le template de départ pour votre application

```
? Choose a template: » - Use arrow-keys. Return to submit.
    ----- Managed workflow -----
>   blank                 a minimal app as clean as an empty canvas
    blank (TypeScript)    same as blank but with TypeScript configuration
    tabs (TypeScript)     several example screens and tabs using react-navigation and TypeScript
    ----- Bare workflow -----
    minimal               bare and minimal, just the essentials to get you started
    minimal (TypeScript)  same as minimal but with TypeScript configuration
```

Choisissez blank pour avoir un projet vierge.

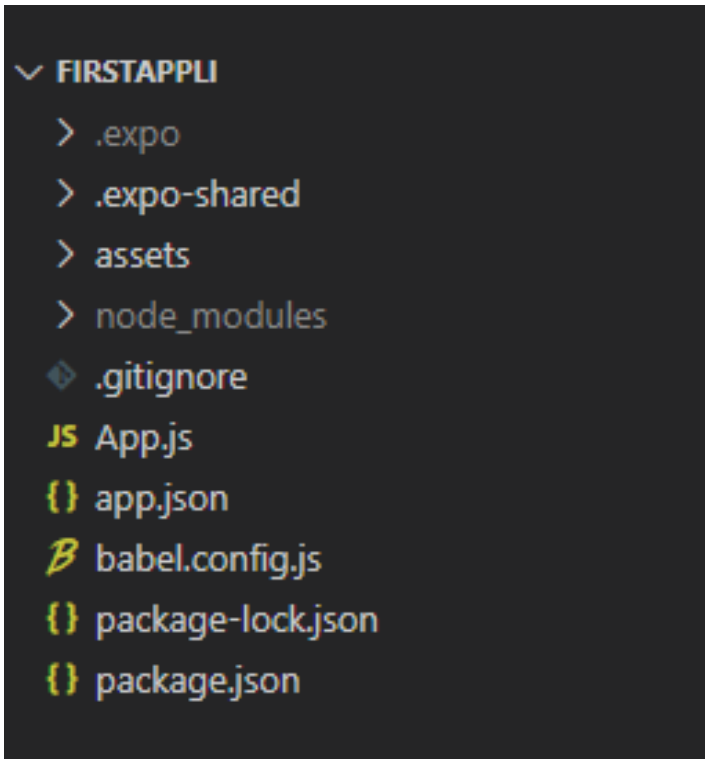
Après quelques installations vous devrez avoir ceci

```
✔ Choose a template: » blank                a minimal app as clean as an empty canvas
✔ Downloaded and extracted project files.
› Using npm to install packages.
✔ Installed JavaScript dependencies.

› Your project is ready!

To run your project, navigate to the directory and run one of the following npm commands.

- cd FirstAppli
- npm start # you can open iOS, Android, or web from here, or run them directly with the commands below.
- npm run android
- npm run ios # requires an iOS device or macOS for access to an iOS simulator
- npm run web
```

Voilà vous avez créé votre application,mettez vous dans le dossier de l'application, lancez votre application avec `npm start` dans l'invite de commande et vous aurez un QR code a scanné avec l'application Expo Go.

**Attention !!!!!!!!** Votre téléphone devra être sur le même réseau que la ou vous ferez npm start

Pour l'instant votre application n'est constituée de rien a pars un message, ouvrez votre éditeur de code et ajoutez le dossier du projet dans l'éditeur. En ouvrant votre éditeur de code vous aurez accès à ça :




Dans tous ces fichiers/dossiers, nous n'avons besoin que de App.js qui est la première page sur laquelle l'application est lancée. Cliquez dessus et vous pourrez commencer à développer.

Arrivé sur App.js nous avons ceci

```
import { StatusBar } from 'expo-status-bar';
import React from 'react';
import { StyleSheet, Text, View } from 'react-native';

export default function App() {
  return (
    <View style={styles.container}>
      <Text>Open up App.js to start working on your app!</Text>
      <StatusBar style="auto" />
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#fff',
    alignItems: 'center',
    justifyContent: 'center',
  },
});
```

React Native n'est pas un code intelligent, nous devons lui importer chaque élément dont on a besoin. Comme nous pouvons le voir sur l'image ci-dessus, lors de la création, l'application à importer: StyleSheet (Moyen de faire du Css sur l'application), Text comme nous pouvons le voir cela permet de mettre du text entre les deux balises, sur cet page nous avons aussi le View qui est la mise en forme de l'application. A partir du moment qu'on est dans le "return" ça commencera toujours par une balise View et finira par une balise View. Si vous voulez plus de renseignements pour les composants de React Native, je vous conseille d'aller voir la documentation de React Native ici.

# 2.Commencer à travailler dessus

**Ajout de composant**

Maintenant que nous avons vu comment créer son application et la lancer, nous allons commencer par utiliser des composants(Text, bouton, etc ...)

Pour commencer nous allons créer un formulaire, sans réel code derrière juste pour voir comment marche.

La mise en place des éléments, nous aurons besoin d'importer TextInput comme ceci :

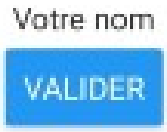
```
import { StyleSheet, Text, View,TextInput } from 'react-native';
```

Et aussi de Button donc nous allons ajouter Button dans cette liste.

Ensuite dans le code nous allons faire ceci :

```
export default function App() {
  return (
    <View style={styles.container}>
      <TextInput>Votre nom</TextInput>
      <Button title="Valider"/>
      <StatusBar style="auto" />
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#fff',
    alignItems: 'center',
    justifyContent: 'center',
  },
});
```

Ce qui nous donne ceci




### Ajout du style sur un composant

Mais le problème que nous avons c’est que nous ne discernons pas bien le TextInput a cause du fond blanc, nous allons donc mettre un peu de style mais que sur le TextInput, comme ceci:

```jsx
import { StatusBar } from "expo-status-bar";
import React from "react";
import { StyleSheet, Text, View, TextInput, Button } from "react-native";

export default function App() {
  return (
    <View style={styles.container}>
      <TextInput style={styles.styleTextInput}>Votre nom</TextInput>
      <Button title="Valider" />
      <StatusBar style="auto" />
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: "#fff",
    alignItems: "center",
    justifyContent: "center",
  },
  styleTextInput: {
    marginBottom: 10,
    height: 45,
    width: "80%",
    borderColor: "#000000",
    borderWidth: 1,
  },
});
```

Si vous avez fait attention, dans la balise TextInput, il y a style qui c'est rajouté, si vous ne le mettez pas alors votre style que vous avez fait en bas ne servira à rien. Dans le styles qu'on peut voir en bas, il y a "styleTextInput" qui a été ajouté, pour ceux qui ont reconnu, le style sur react native est du CSS mais avec quelque différence, suite aux style qui a était ajouté cela nous donne ceci:


Comme vous pouvez le voir, le TextInput a été agrandie que ça soit en longueur ou hauteur, ceci est dû a “height” pour la hauteur et “width” pour la longueur,dans les deux cas pour la valeur nous pouvons mettre un chiffre ou un pourcentage.Maintenez, pour le second changement qu’il y a eu c’est que le TextInput a été espacés du bouton, ce qui est plus agréable en terme de visuel, ce qui nous a permis de le faire c’est le “marginBottom” , cela fait une marge en fonction du composant en dessous du composant dans lequel on a placé le margin, nous avons aussi margin en général qui fait une marge tout autour du composant on a juste besoin de faire “margin”,il y aussi “padding”, qui fait à peu près la même chose mais margin agit sur le tout composant y compris la bordure, alors que padding agit que sur le composant et non sur la bordure.Et pour finir il y a bordure du composant qui est fait avec “borderWidth” qui fait le tour du composant, et “borderColor” qui est la couleur de la bordure. Que sa soit la bordure, la marge ou le padding on peut agir que sur un côté si on le souhaite, comme fait précédemment avec le margin on peut le refaire avec la bordure et le padding

**Gestion des placements des composants**

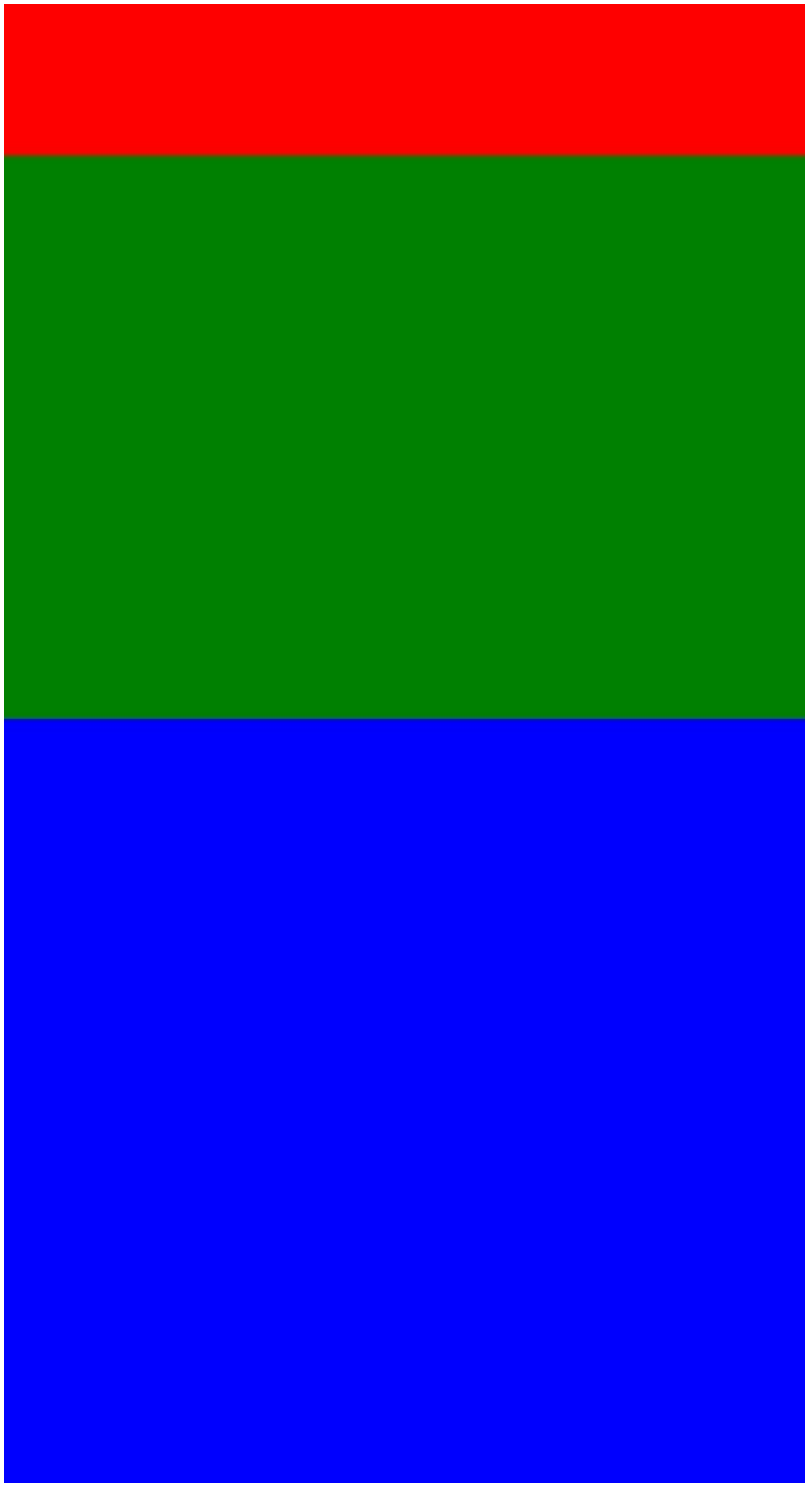
Actuellement tous les composants de votre application sont centrés, ceci est dû aux “justifyContent” et “alignItems” qui sont situées dans “container”.

```
container: {
  flex: 1,
  backgroundColor: "#fff",
  alignItems: "center",
  justifyContent: "center",
},
```

Il n’y a pas que ça, enlever le flex:1 et vous verrez ce qu’il se passe, si vous avez essayé vous avec pus voir que vos composants sont tout en haut de votre application, après vous pouvez mettre un height sur votre container mais la aussi, cela ne marchera pas trés bien, car cela marchera pour votre téléphone mais si vous prenez un téléphone plus petit ou plus grand vous verrez que la disposition n’ai pas la même.Nous allons donc utilisez Flex pour placer ces composants, flex gardera la même vue sur n’importe quel autre pareil. Pour mieux comprendre flex, en faite si vous mettez flex 1 cela va prendre 1/1 sur l’écran donc tout l’écran. En modifiant un peu le code pour vous montrer comment marche les flex supérieur à 1 :

```
import { StatusBar } from "expo-status-bar";
import React from "react";
import { StyleSheet, Text, View, TextInput, Button } from "react-native";

export default function App() {
  return (
    <View style={{ flex: 1, backgroundColor: "yellow" }}>
      <View style={{ flex: 1, backgroundColor: "red" }}></View>
      <View style={{ flex: 2, backgroundColor: "green" }}></View>
      <View style={{ flex: 3, backgroundColor: "blue" }}></View>
    </View>
  );
```

Nous obtenons ceci



On peut ainsi voir que le flex a 3 est plus grand que les autres flex car nous sommes sur une View a flex 1 et sur cette vue nous demandons au view bleu qu'il prend 3/6 car notre application est coupée en 6 espaces, car l'élément rouge est le premier composant , le vert est deuxième et le bleu est troisième donc 1+2+3= 6, le composant vert va prendre 2/6 de l'écran et le rouge 1/6.

Maintenant qu'on a vu les flex, on va parler de flexDirection sur flex direction il y a 4 options: column, row, column-reverse ,row-reverse. Le mieux pour que vous comprenez sera de scanner ce QR code avec expo GO:

Faites des tests avec les options et vous verrez leurs effets que cela donne.Maintenant que nous avons vu comment marche les flex. Nous sommes capable de savoir comment positionner nos composants et de rendre un visuel plus agréable et éviter que nos composants sont bloqués en haut.

> On ne peut pas modifier le style d'un bouton, vous pouvez essayer et cela ne changera pas le bouton

**Récupérer la donnée d'un TextInput**

Maintenant que nous savons comment mettre un style sur nos composant, on va commencer par récupérer une donnée qu'on va mettre dans le TextInput et l'afficher dans une balise Text, on va commencer par faire un constructeur:

```
class App extends React.Component {

 constructor(props) {
    super(props);

    this.state = {
      nom:''
    }
  }
```

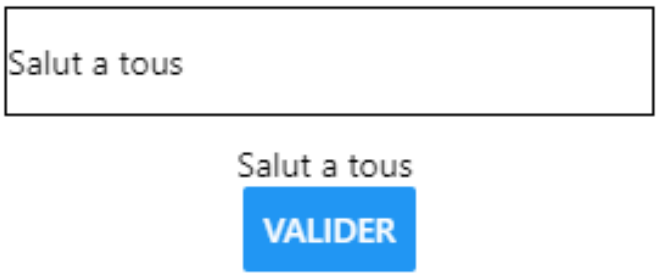
Maintenant que le constructeur est fait on va dire au TextInput de récupérer ce qu'on écrit dedans

```
    return (
      <View style={styles.container}>
      <TextInput style={styles.styleTextInput}
      placeholder="Votre nom"
      onChangeText={(text) => this.setState({ nom: text })}
      />
      <Button title="Valider" />
    </View>
    );
  }
}
```

Et on va créer une balise Text et lui dire que les données qu'il va afficher sera la donnée écrite par le TextInput.

```
<Text>{this.state.nom}</Text>
```

Maintenant que ça a été fait, écrivez dans le TextInput et vous verrez qu'au dessus du bouton un text apparaitra.




Maintenant que nous savons comment ça marche pour récupérer une donnée d'un TextInput, nous allons nous occuper du bouton, et de faire une action quand on clique dessus. On va faire apparaître une alerte au moment qu'on clique sur le bouton.Donc on doit importer Alert

```
import {
  StyleSheet,
  View,
  Text,
  TextInput,
  Button,
  Alert,
} from "react-native";
```

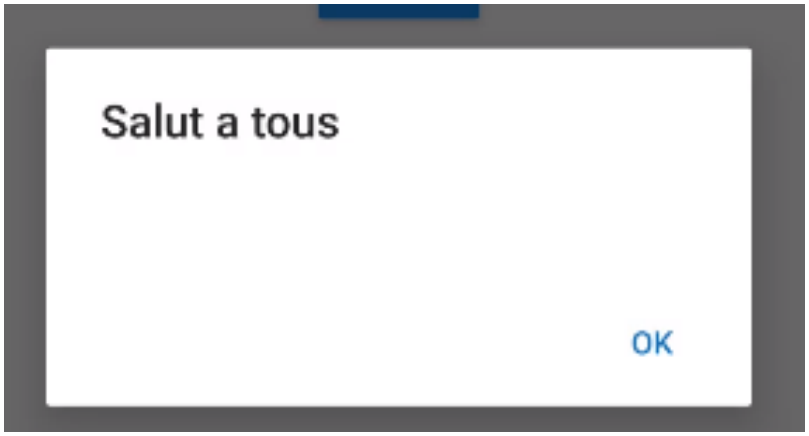
Puis créer une fonction dans le vue qui va être appelé lors de l'appuie sur le bouton et y mettre l'alerte dedans

```
  _Valider(){
    Alert.alert(this.state.nom)

  }
```

Une fois qu'on a fais ça, on va faire dire au bouton d'appeler la fonction

```
render() {

  return (
    <View style={styles.container}>
    <TextInput style={styles.styleTextInput}
    placeholder="Votre nom"
    onChangeText={(text) => this.setState({ nom: text })}
    />
    <Button title="Valider" onPress={() => {this._Valider()}} />
  </View>
  );
}
}
```

Une fois ceci fait, lancer votre application sur votre téléphone et essayer par vous même.Ce qui vous donnera ceci:



**Navigation entre différente vue**

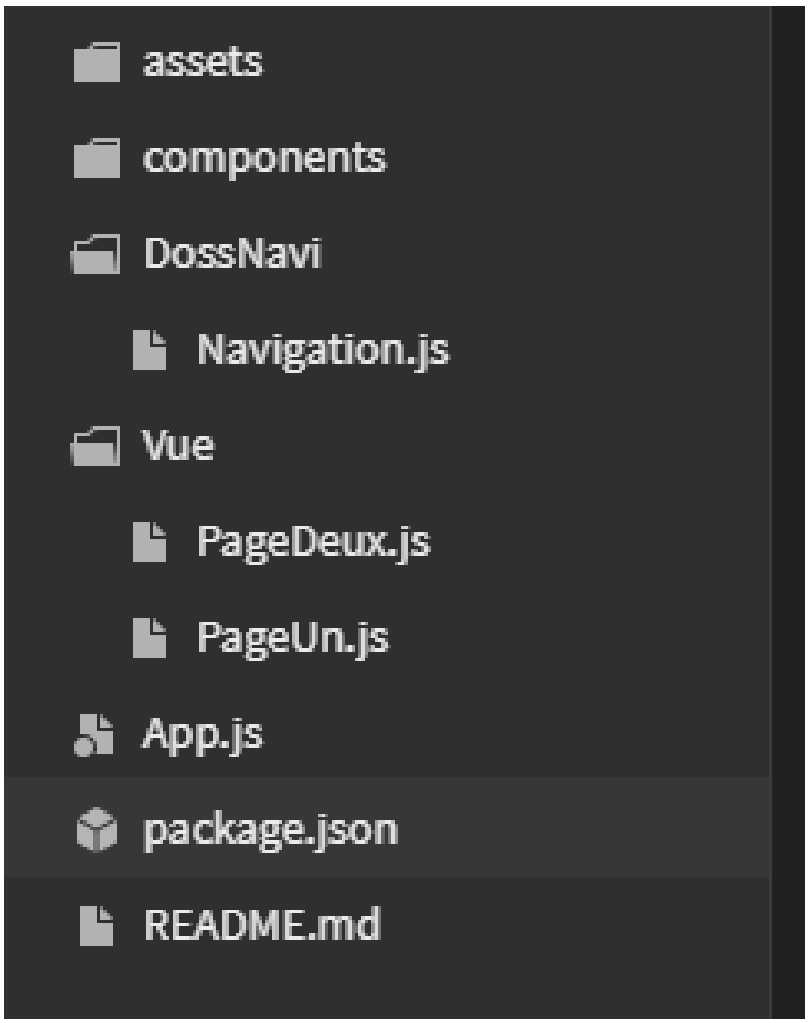
Pour commencer éteignez votre application dans l'invite de commande avec CTRL+C puis écrivez:

```
npm install --save react-navigation
```

Après l'installation vous pouvez relancer l'application avec

```
npm start
```

Pour commencer votre navigation, vous devez créer des dossiers et des fichiers en .js




Comparer à avant il y a le dossier DossNavi et le dossier Vue, une fois que vous avez les fichiers que vous venez de créer mettais ce code dans PageUn.js

```
import React from "react";
import {
  StyleSheet,
  View,
  Text,
  TextInput,
  Button,
  Alert,
} from "react-native";

class PageUn extends React.Component {

  _ChangeVue(){
   this.props.navigation.navigate("PageDeux");

  }

  render() {

    return (
      <View style={styles.container}>
      <Button title="Vers la page suivante !" onPress={() => {this._ChangeVue()}} />
    </View>
    );
  }
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: "#fff",
    alignItems: "center",
    justifyContent: "center",
  },

});

export default PageUn;
```

Et dans PageDeux.js:

```
import React from "react";
import {
  StyleSheet,
  View,
  Text,
  TextInput,
  Button,
  Alert,
} from "react-native";

class PageDeux extends React.Component {

  render() {

    return (
      <View style={styles.container}>
     <Text> Bienvenue sur la page deux</Text>
    </View>
    );
  }
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: "#fff",
    alignItems: "center",
    justifyContent: "center",
  },
});

export default PageDeux;
```

Dans import j'ai laisser les composant qu'on a utilisée avant, si jamais on en a besoin plus tard ça évitera de les remettre

Ensuite dans Navigation.js

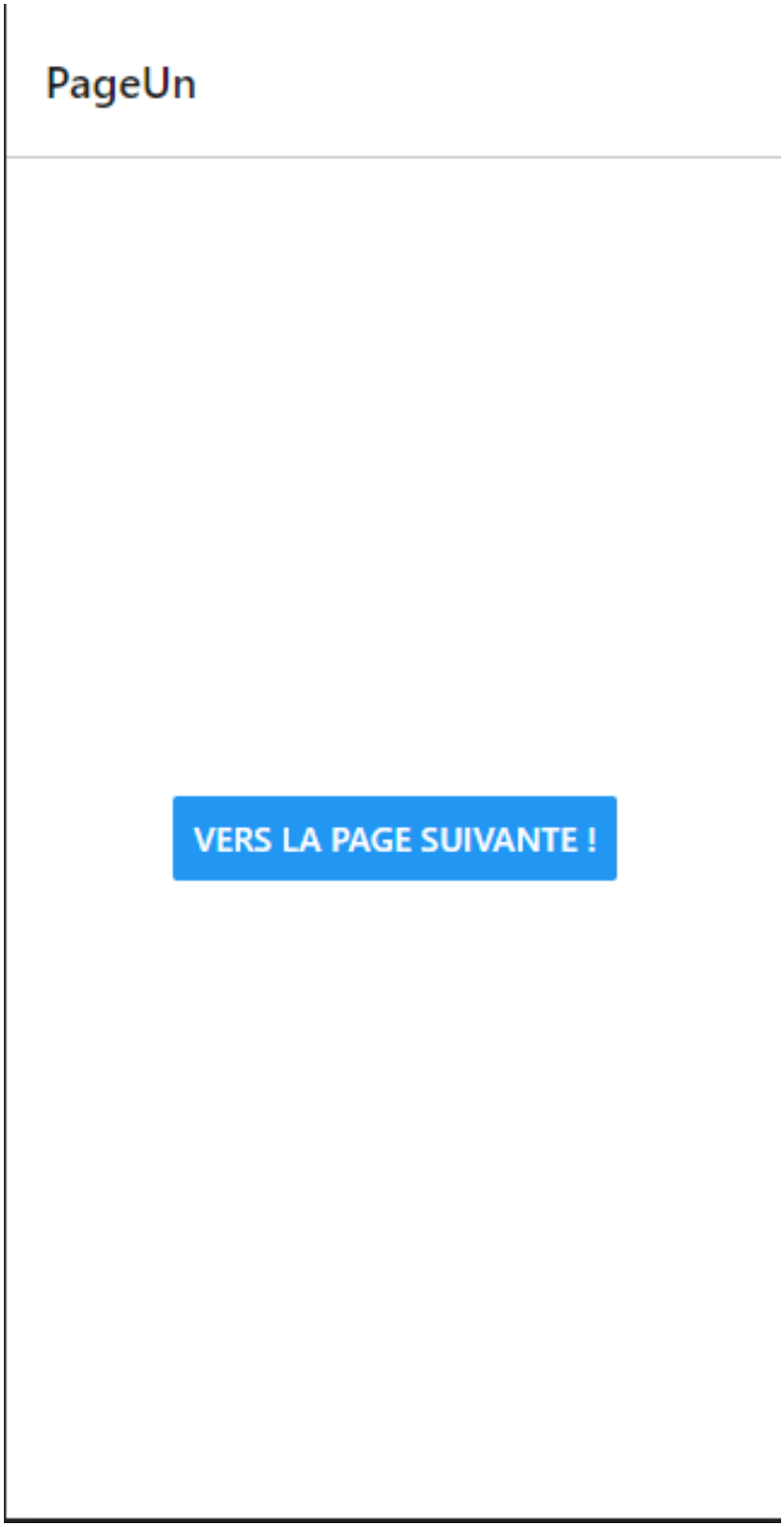
```
import * as React from "react";
import { createStackNavigator } from "@react-navigation/stack";
import PageDeux from '../Vue/PageDeux';
import PageUn from '../Vue/PageUn';

const Stack = createStackNavigator();

function MyStack() {
  return (
    <Stack.Navigator>
        <Stack.Screen name="PageUn" component={PageUn} />
        <Stack.Screen name="PageDeux" component={PageDeux} />
    </Stack.Navigator>
  )}

  export default MyStack;
```

Et pour finir dans App.js

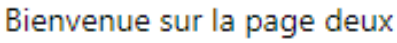
```
import React from "react";
import { NavigationContainer } from "@react-navigation/native";
import Mystack from "./DossNavi/Navigation";

export default function App() {
  return (
    <NavigationContainer>
      <Mystack />
    </NavigationContainer>
  );}

```

Quand vous allez lancer votre application, la première page qui va se lancer est App.js donc dans App.js, on importe MyStack de la navigation et on l'appel aux moment du return , NavigationContainer permet de gérer votre navigation.

Si vous avez fait comme comme au dessus normalement vous arriverez sur ça quand vous lancerez l'application :




Quand vous cliquerez sur le bouton vous changerez de page pour arrivez sur la page deux


Voilà vous avez réussi votre navigation. Maintenant vous pouvez personnaliser votre navigation en suivant cette doc

#### Faire une liste et la remplir

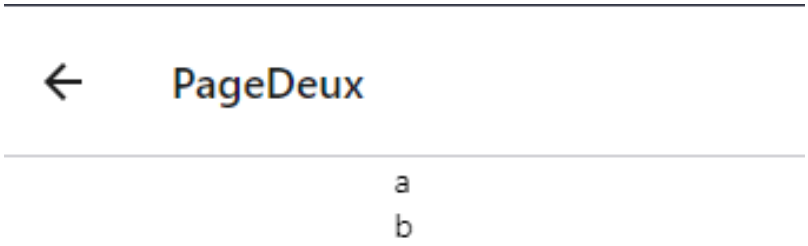
Sur PageDeux.js que nous venons de créer on va établir une liste et la remplir de donnée. Pour ce faire nous avons besoin de FlatList donc nous allons importer FlatList comme ceci

```
import {
  StyleSheet,
  View,
  Text,
  TextInput,
  Button,
  Alert,
  FlatList,
} from "react-native";
```

Après nous allons remplacer le texte par FlatList et faire ceci :

```
class PageDeux extends React.Component {

  render() {

    return (
    <View style={styles.container}>
     <FlatList
        data={[{key: 'a'}, {key: 'b'}]}
        renderItem={({item}) => <Text>{item.key}</Text>}
     />
    </View>
    );
  }
}
```

Maintenant si vous allez sur la deuxième page avec votre téléphone vous aurez ceci



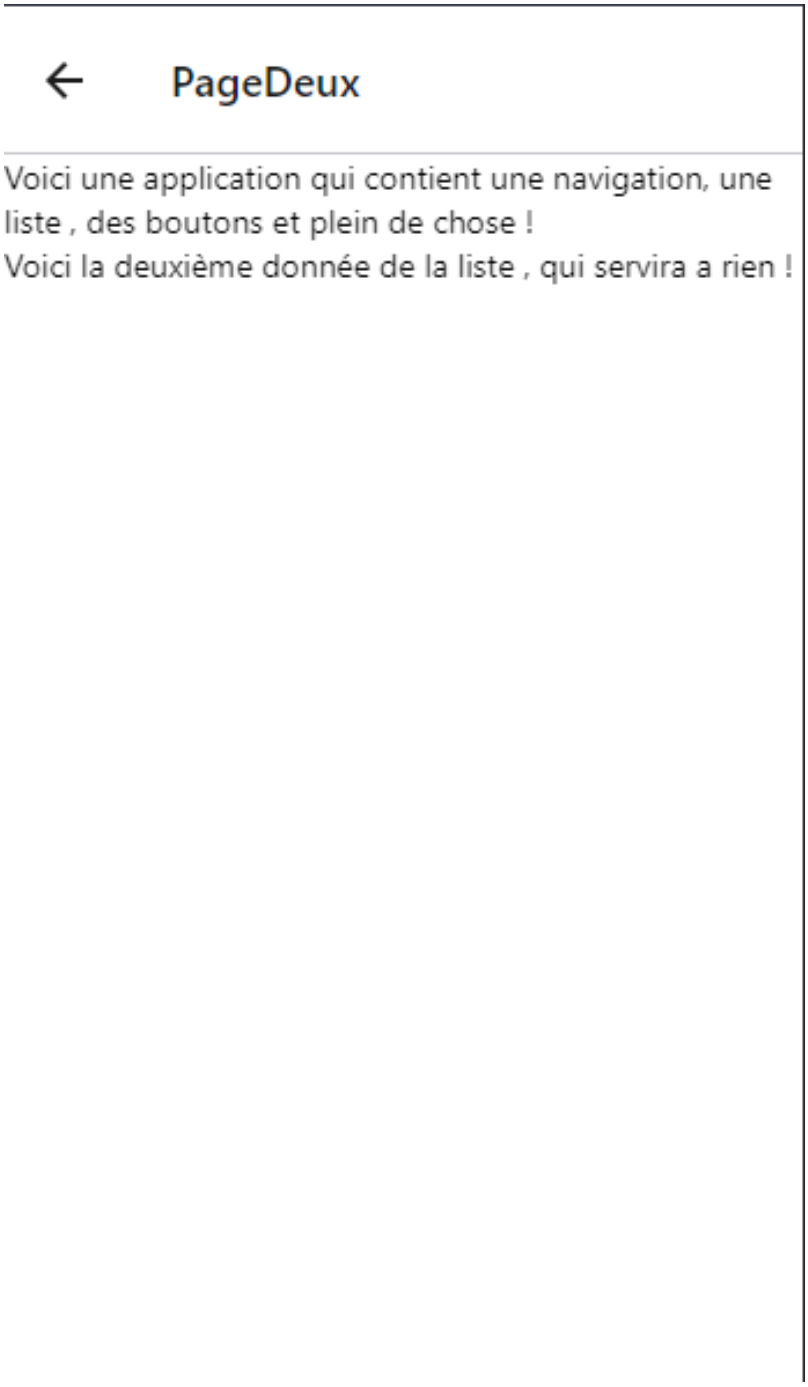

Voici votre première liste mais, elle n'est pas complète si on veut une liste complète avec plein de détails cela risque d'être compliqué. on va créer un dossier Data appeler le comme vous le voulez, tant qu'il est un minimum de sens.Une fois fait, créez y un fichier en .js :

```
export default data = [
    {
        id:1,
        desc:"Voici une application qui contient une navigation, une liste , des boutons et plein de chose !"

    },
    {
        id:2,
        desc:"Voici la deuxième donnée de la liste , qui servira a rien !"

    }
]
```

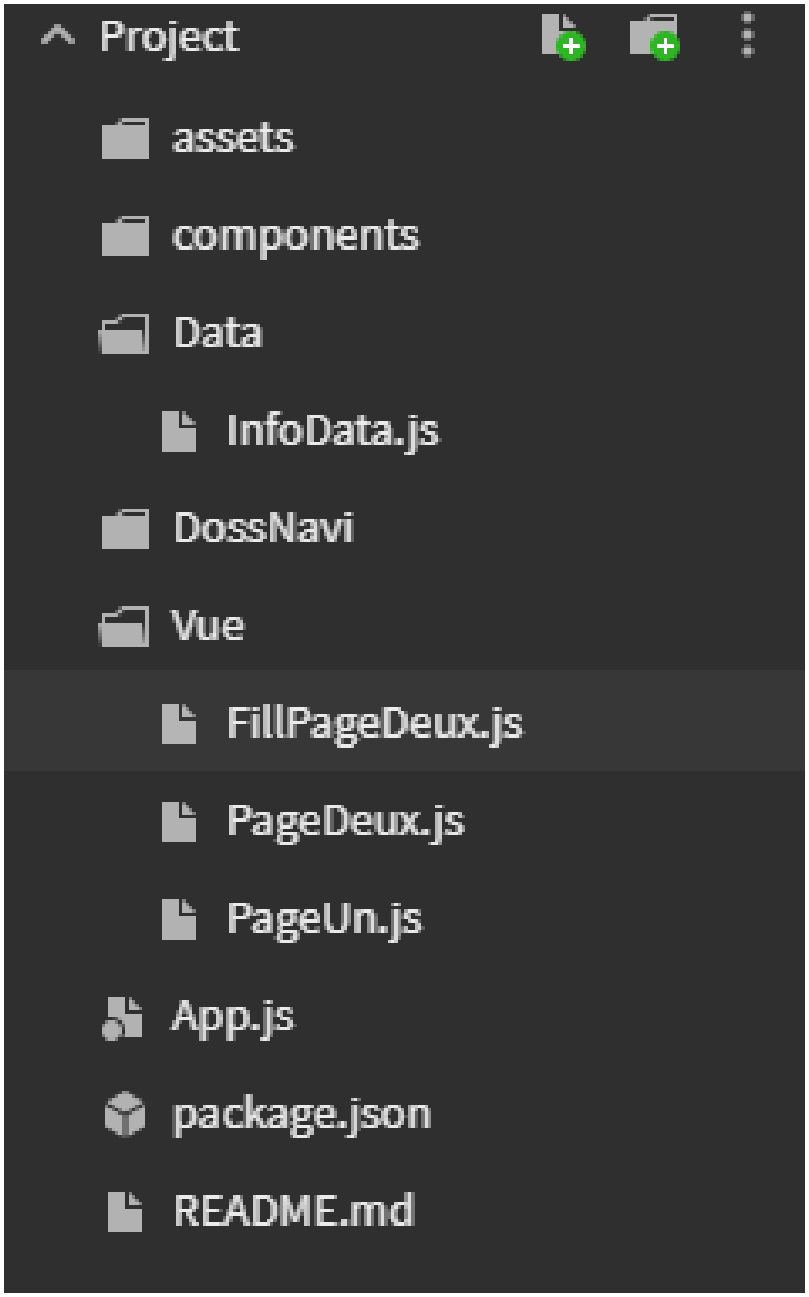
Ensuite dans le fichier ou il y a la FlatList on doit importer le fichier qui contient les données ce qui nous donne ceci :

```jsx
import React from "react";
import {
  StyleSheet,
  View,
  Text,
  TextInput,
  Button,
  Alert,
  FlatList,
} from "react-native";
import infodata from '../Data/InfoData'

class PageDeux extends React.Component {

  render() {

    return (
    <View style={styles.container}>
     <FlatList
        data={infodata}
        renderItem={({item}) => <Text>{item.desc}</Text>}
     />
    </View>
    );
  }
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: "#fff",
    alignItems: "center",
    justifyContent: "center",
  },
});

export default PageDeux;
```

et dans l'application cela donne:




Il y a un problème qu'on peut constater, il y a aucun visuel et on ne va pas s'amuser à faire tout un style dans la FlatList, donc nous allons créer une page à côté, personnellement je commence les pages qu'on va mettre dans la FlatList par Fill (remplir) car pour moi cette page va servir à alimenter la FlatList.




Ensuite le code qu'on va mettre dedans sera :

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

class FillPageDeux extends React.Component {


  render() {
    const data= this.props.data
    return (
      <View style={styles.container}>
      <Text style={styles.textStyle}>{data.id} </Text>
      <Text style={styles.textStyle}>{data.desc} </Text>
    </View>
    );
  }
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: "#958686",
    alignItems: "center",
    justifyContent: "center",
    margin:20,
    borderRadius:10,
  },
  textStyle:{
    margin:10,
  }

});

export default FillPageDeux;

```

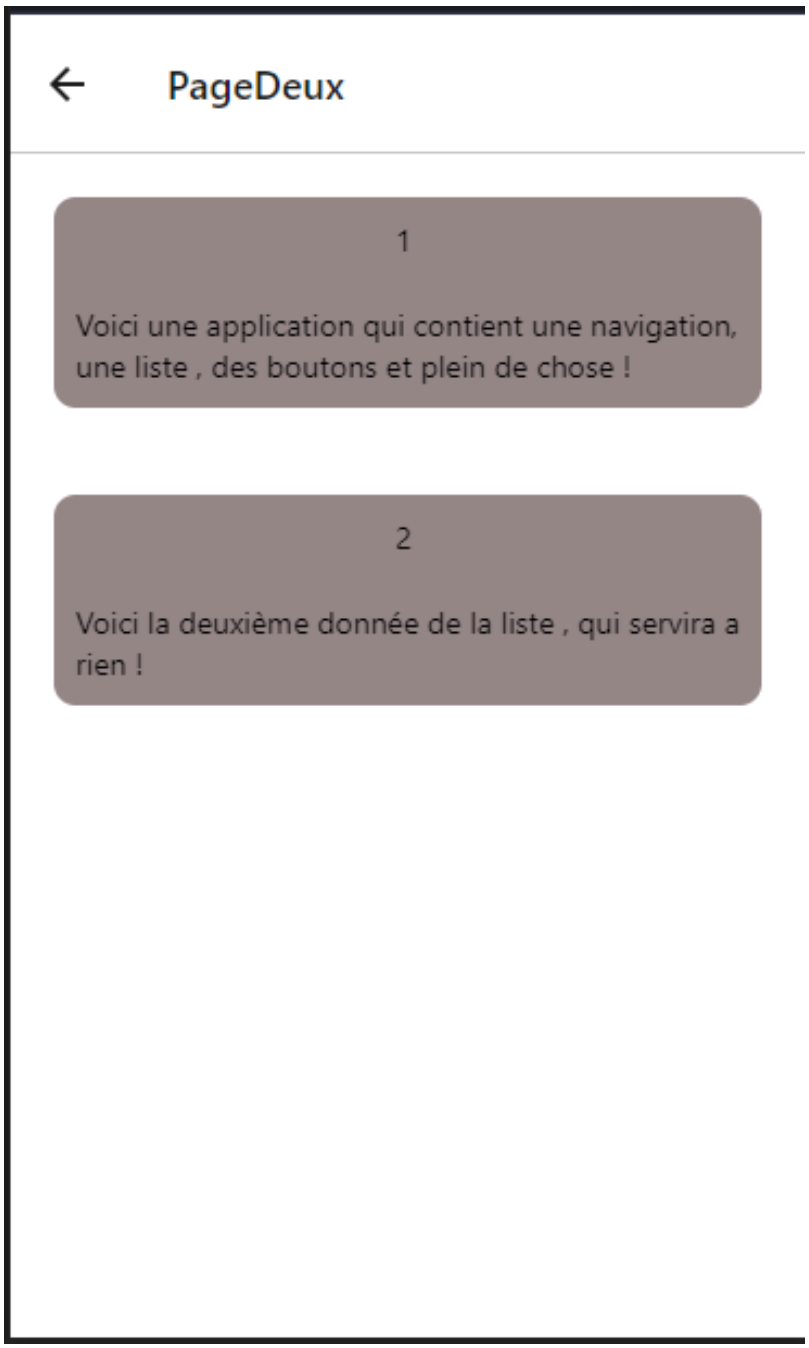
Maintenant il faut importer cette page vers la FlatList

```
import FillPageDeux from './FillPageDeux';
```

Et ensuite modifier le renderItem de la FlatList

```
<FlatList
    data={infodata}
    renderItem={({item}) => <FillPageDeux
    data={item}
    />}
/>
```

Ce qui vous permettra d'avoir




Vous avez réussi à faire un FlatList Voilà vous avez fait votre première application