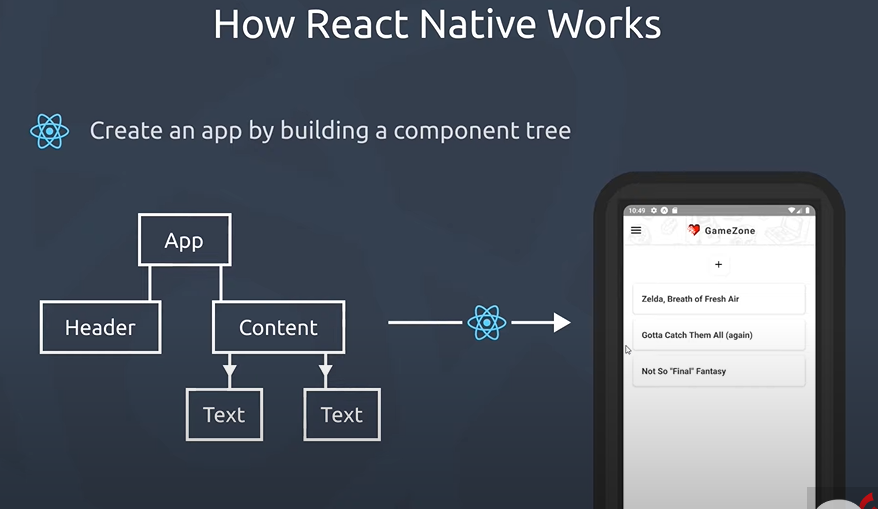
React Native

1. Introduction
   1. Introduction

* Use React to create mobile apps (Android and Ios)
* Great choice for developers already familiar with React for web
  1. How React Native works



React natives provides us with a bunch of components out of the box that we can use to make our own layouts , and when we build or test our react native app react native takes this components and I compiles them into native code and widgets that work on android and ios that then we can see them in either our real device or some kind of emulator now

* 1. What we need to install
* Node
* Android Studio

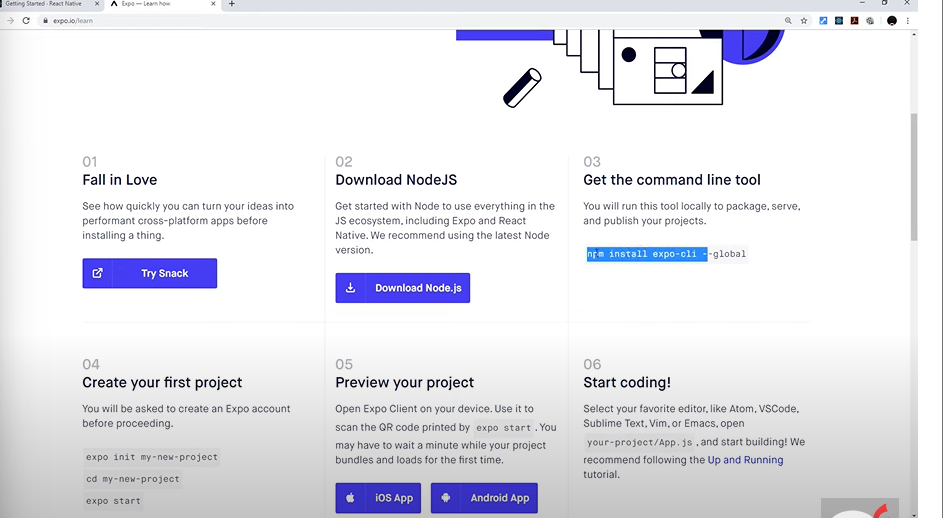
1. Creating a react Native App
   1. Thing that we need to do

We will use Expo because we are beginner

Une image contenant texte

Description générée automatiquement

We will use Expo CLi because we are beginner



Android Studio is done We need just to add a virtual device manager.

1. Basic component in React Native & Style
   1. Views, Text & Styles

* View
  + == Div (We use it to wrap a component)
* Text
  + (We use it every time we want to output a bit of text)
* StyleSheet
  + React Native don’t use css it just emulates the idea of css because Android and iOS don’t support css so for css we use StyleSheet to create our css.
* import { StyleSheet, Text, View } from "react-native";
* export default function App() {
* return (
* <View style={styles.container}>
* <View style={styles.header}>
* <Text style={styles.boldText}>Hello ,World</Text>
* </View>
* <View style={styles.body}>
* <Text style={styles.boldText}>
* Lorem ipsum <Text>Test</Text> dolor sit amet.
* </Text>
* <Text>Lorem ipsum dolor sit amet.</Text>
* <Text>Lorem ipsum dolor sit amet.</Text>
* </View>
* </View>
* );
* }
* const styles = StyleSheet.create({
* container: {
* flex: 1,
* backgroundColor: "#fff",
* alignItems: "center",
* justifyContent: "center",
* },
* header: {
* backgroundColor: "pink",
* padding: 20,
* },
* boldText: {
* fontWeight: "bold",
* },
* body: {
* backgroundColor: "yellow",
* padding: 20,
* },
* });



* 1. Using state
     1. using State using onPress

import { useState } from "react";

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

export default function App() {

  const [name, setName] = useState("shaun");

  const [person, setPerson] = useState({ name: "mario", age: 40 });

  const clickHandler = () => {

    setName("chun-li");

    setPerson({ name: "Houssam", age: 45 });

  };

  return (

    <View style={styles.container}>

      <Text>My name is {name}</Text>

      <Text>

        His name is {person.name} and his age is {person.age}

      </Text>

      <View style={styles.buttonContainer}>

        {/\* In React Native we cant use Button like this  \*/}

        {/\* <Button>Click Me</Button> \*/}

        {/\* We can't style the Button omponent but wa have a solution we cane

           style It using the view \*/}

        <Button title="update state" onPress={clickHandler} />

      </View>

    </View>

  );

}

const styles = StyleSheet.create({

  container: {

    flex: 1,

    backgroundColor: "#fff",

    alignItems: "center",

    justifyContent: "center",

  },

  header: {

    backgroundColor: "pink",

    padding: 20,

  },

  boldText: {

    fontWeight: "bold",

  },

  body: {

    backgroundColor: "yellow",

    padding: 20,

  },

  buttonContainer: {

    marginTop: 20,

  },

});

Une image contenant texte

Description générée automatiquement

* + 1. Using state using onChangeText on an Input

import { useState } from "react";

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

export default function App() {

  const [name, setName] = useState("shaun");

  const [age, setAge] = useState("30");

  return (

    <View style={styles.container}>

      <Text>EnterName:</Text>

      {/\* We need to add a css to our TextInput  because there is no css \*/}

      <TextInput

        style={styles.input}

        placeholder="Enter your name"

        onChangeText={(val) => setName(val)}

        // multiline helps allow us to dd multiple different line inside text Input if we press enter

        multiline

      />

      <Text>Enter age:</Text>

      <TextInput

        style={styles.input}

        placeholder="Enter your age"

        onChangeText={(val) => setAge(val)}

        // the keyboard that pops up is now numeric

        keyboardType="numeric"

      />

      {/\* You can see all the different TextInput props in ReactNAtive web Site\*/}

      <Text>

        name:{name},age:{age}

      </Text>

    </View>

  );

}

const styles = StyleSheet.create({

  container: {

    flex: 1,

    backgroundColor: "#fff",

    alignItems: "center",

    justifyContent: "center",

  },

  input: {

    borderWidth: 1,

    borderColor: "#777",

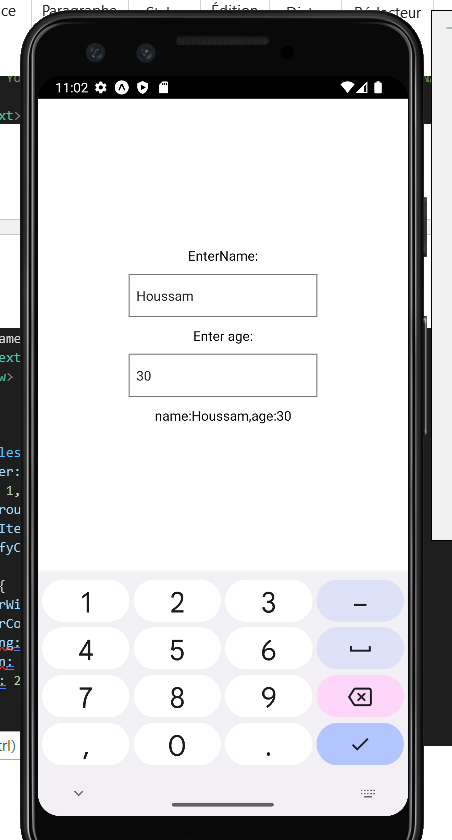
    padding: 8,

    margin: 10,

    width: 200,

  },

});



* 1. Using Lists & ScrollView

import { useState } from "react";

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

export default function App() {

  const [people, setPeople] = useState([

    { name: "Houssam", key: "1" },

    { name: "Shau", key: "2" },

    { name: "Mario", key: "3" },

    { name: "Luigui", key: "4" },

    { name: "Toad", key: "5" },

    { name: "Boaer", key: "6" },

    { name: "Jose", key: "7" },

    { name: "Mickel", key: "8" },

  ]);

  return (

    <View style={styles.container}>

      {/\* scrollView helps us to create a view with the scroll on ouor wrap \*/}

      <ScrollView>

        {people.map((item) => {

          return (

            <View key={item.key}>

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

            </View>

          );

        })}

      </ScrollView>

    </View>

  );

}

const styles = StyleSheet.create({

  container: {

    flex: 1,

    backgroundColor: "#fff",

    alignItems: "center",

    justifyContent: "center",

  },

  item: {

    marginTop: 24,

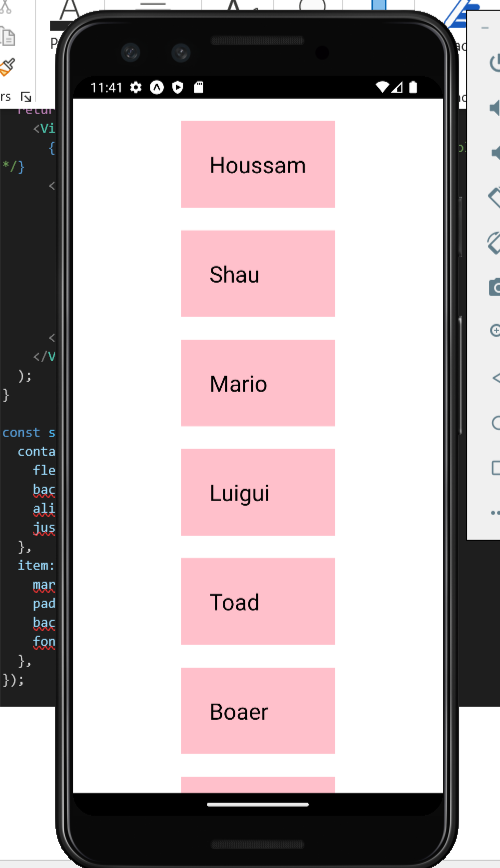
    padding: 30,

    backgroundColor: "pink",

    fontSize: 24,

  },

});



* 1. Flat List
* **Flat List is better for performance.**
* **Flat List looks automatically for the key property on each item**
* **if we don't have key i our item we need to add keyExtractor props in our Flat List component ,keyExtractor props is equal to a function that return what property on the objects is going to use as the key**
* **With Flat List if we have a large array of items ,all the item do not automatically load onto the screen wait first render only the first few and then more we load as we scroll down**
* **Flat List allows as to add numColumns props that helps us to split this out into several different columns**

import { useState } from "react";

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

export default function App() {

  const [people, setPeople] = useState([

    { name: "Houssam", id: "1" },

    { name: "Shau", id: "2" },

    { name: "Mario", id: "3" },

    { name: "Luigui", id: "4" },

    { name: "Toad", id: "5" },

    { name: "Boaer", id: "6" },

    { name: "Jose", id: "7" },

    { name: "Mickel", id: "8" },

  ]);

  return (

    <View style={styles.container}>

      {/\* FlatList is better for performance \*/}

      {/\* FlatList looks automatically for the key property on each item\*/}

      {/\* if we don't have key i our item we need to add keyExtractor props in our

      FlatList component ,keyExtractor props is equal to a function that return

      what property on the objects is going to used as the key \*/}

      {/\* With FlatList if we have a large array of items ,all the item

      do not automatically load onto the scree wait first render only the first

       few and then more we load as we scrool down \*/}

      {/\* FlatList allows as to add numColumns props that helps us to split this out into

       several different columns  \*/}

      <FlatList

        data={people}

        renderItem={({ item }) => <Text style={styles.item}>{item.name}</Text>}

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

        numColumns={2}

      />

    </View>

  );

}

const styles = StyleSheet.create({

  container: {

    flex: 1,

    backgroundColor: "#fff",

    alignItems: "center",

    justifyContent: "center",

  },

  item: {

    marginTop: 24,

    padding: 30,

    backgroundColor: "pink",

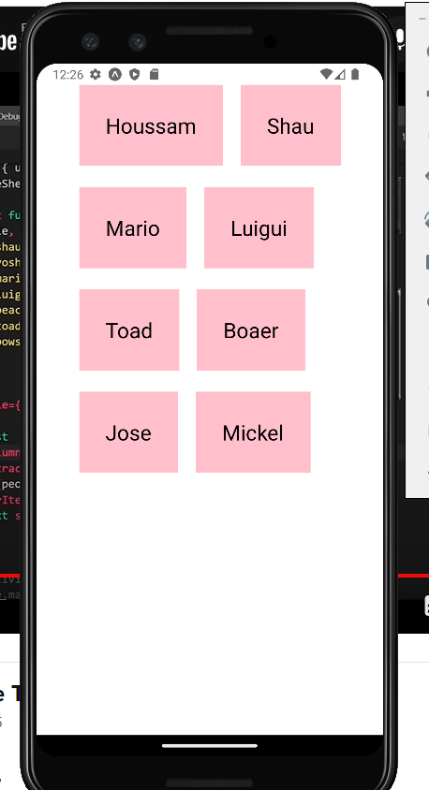
    fontSize: 24,

    marginHorizontal: 10,

    marginTop: 24,

  },

});



* 1. Touchable Component
* We can do OnPress to a Text component And we can't add a style to a Button component so what is the solution ?????
  + So, the Solution is a Touchable component.
  + TouchableOpacity we can add style.

import { useState } from "react";

import {

  FlatList,

  StyleSheet,

  Text,

  TouchableOpacity,

  View,

} from "react-native";

export default function App() {

  const [people, setPeople] = useState([

    { name: "Houssam", id: "1" },

    { name: "Shau", id: "2" },

    { name: "Mario", id: "3" },

    { name: "Luigui", id: "4" },

    { name: "Toad", id: "5" },

    { name: "Boaer", id: "6" },

    { name: "Jose", id: "7" },

    { name: "Mickel", id: "8" },

  ]);

  const pressHandler = (id) => {

    console.log(id)

    setPeople((prevPeople)=>{

      return prevPeople.filter(person=>person.id!=id)

    })

  };

  return (

    <View style={styles.container}>

      {/\* We can do OnPress to a Text component And we can't add a style

        to a Button component so what is the dolution ?????  \*/}

      {/\* So the Solution is a Touchable component \*/}

      {/\* TouchableOpacity we can add style \*/}

      <FlatList

        data={people}

        renderItem={({ item }) => (

          <TouchableOpacity onPress={() => pressHandler(item.id)}>

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

          </TouchableOpacity>

        )}

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

        numColumns={2}

      />

    </View>

  );

}

const styles = StyleSheet.create({

  container: {

    flex: 1,

    backgroundColor: "#fff",

    alignItems: "center",

    justifyContent: "center",

  },

  item: {

    marginTop: 24,

    padding: 30,

    backgroundColor: "pink",

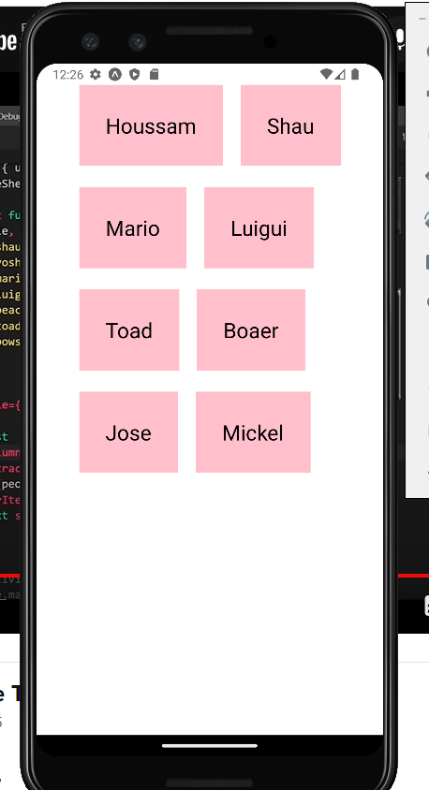
    fontSize: 24,

    marginHorizontal: 10,

    marginTop: 24,

  },

});



On touche un item il vas être supprimer

* 1. Alert

const submitHandler = (text) => {

    if (text.length > 3) {

      setTodos((prevTodos) => {

        return [...prevTodos, { text: text, key: Math.random().toString() }];

      });

    } else {

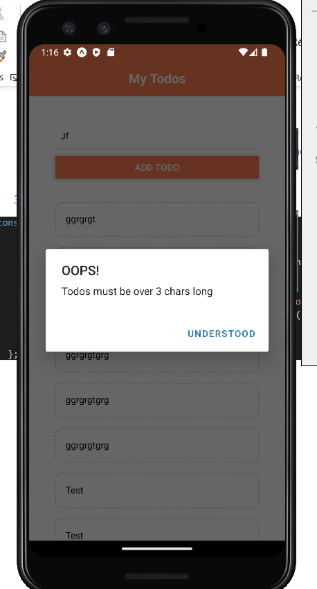
      Alert.alert("OOPS!", "Todos must be over 3 chars long", [

        { text: "Understood", onPress: () => console.log("alert closed") },

      ]);

    }

  };



* 1. Dismissing the keyboard

Helps us if we click away from the keyboard, It should close the keyboard

<TouchableWithoutFeedback

      onPress={() => {

        Keyboard.dismiss();

        console.log("dimissed keyboard");

      }}

    >

.

.

.

</TouchableWithoutFeedback>

* 1. FlexBox

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

export const SandBox = () => {

  return (

    <View style={styles.container}>

      <Text style={styles.boxOne}>One</Text>

      <Text style={styles.boxTwo}>Two</Text>

      <Text style={styles.boxThree}>Three</Text>

      <Text style={styles.boxFour}>Four</Text>

    </View>

  );

};

const styles = StyleSheet.create({

  container: {

    flexDirection: "row",

    justifyContent: "space-around",

    alignItems: "flex-end",

    paddingTop: 40,

    backgroundColor: "#ddd",

  },

  boxOne: {

    backgroundColor: "violet",

    padding: 10,

  },

  boxTwo: {

    backgroundColor: "gold",

    padding: 20,

  },

  boxThree: {

    backgroundColor: "coral",

    padding: 30,

  },

  boxFour: {

    backgroundColor: "skyblue",

    padding: 40,

  },

});

Une image contenant texte

Description générée automatiquement

container: {

    flexDirection: "row",

    justifyContent: "space-around",

    alignItems: 'center',

    paddingTop: 40,

    backgroundColor: "#ddd",

  },

Une image contenant texte, capture d’écran, moniteur

Description générée automatiquement

const styles = StyleSheet.create({

  container: {

    flexDirection: "row",

    justifyContent: "space-around",

    alignItems: "center",

    paddingTop: 40,

    backgroundColor: "#ddd",

  },

  boxOne: {

    flex: 2,

    backgroundColor: "violet",

    padding: 10,

  },

  boxTwo: {

    flex: 1,

    backgroundColor: "gold",

    padding: 20,

  },

  boxThree: {

    flex: 1,

    backgroundColor: "coral",

    padding: 30,

  },

  boxFour: {

    flex: 1,

    backgroundColor: "skyblue",

    padding: 40,

  },

});

Une image contenant texte, capture d’écran, moniteur

Description générée automatiquement

* 1. Projecr One

In this Project we have all precedent things that we learn

This is the Link

1. Project Two



* 1. Custom Fonts using Expo.

We download the Font what we need from google fonts and we place it in assets/fonts.

App.js

export default function App() {

  useEffect(() => {

    Font.loadAsync({

      "nunito-regular": require("./assets/fonts/Nunito-Regular.ttf"),

      "nunito-bold": require("./assets/fonts/Nunito-Bold.ttf"),

    });

  },[]);

  return (

    <View style={styles.container}>

      <Home />

    </View>

  );

}

Home.js

export const Home = () => {

  return (

    <View style={styles.container}>

      <Text style={styles.titleText} >Home Screen</Text>

    </View>

  );

};

const styles = StyleSheet.create({

  container: {

    padding: 24,

  },

  titleText:{

    fontFamily:'nunito-bold'

  }

});

* 1. Global Styles

Styles/styles.js

import { StyleSheet } from "react-native";

export const globalStyles = StyleSheet.create({

  container: {

    flex: 1,

    padding: 24,

  },

  titleText: {

    fontFamily: "nunito-bold",

    fontSize:18,

    color:'#333'

  },

  paragraph:{

    marginVertical:8,

    lineHeight:20,

  }

});

Home.js

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

import { globalStyles } from "../styles/styles";

export const Home = () => {

  return (

    <View style={globalStyles.container}>

      <Text style={globalStyles.titleText}>Home Screen</Text>

    </View>

  );

};

* 1. Navigate

Navigate helps us to navigate between our components.

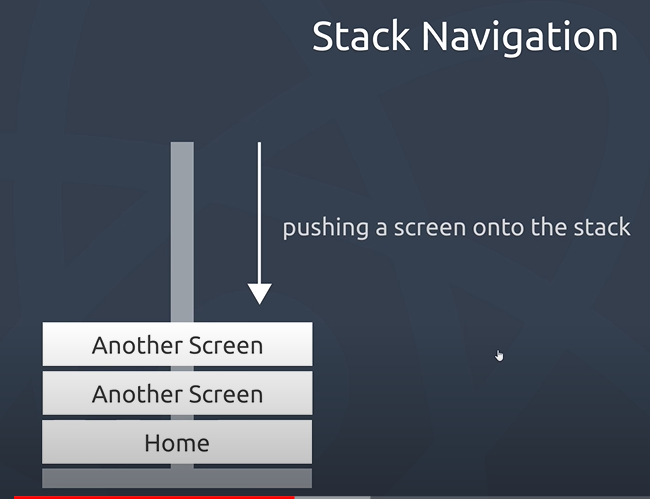
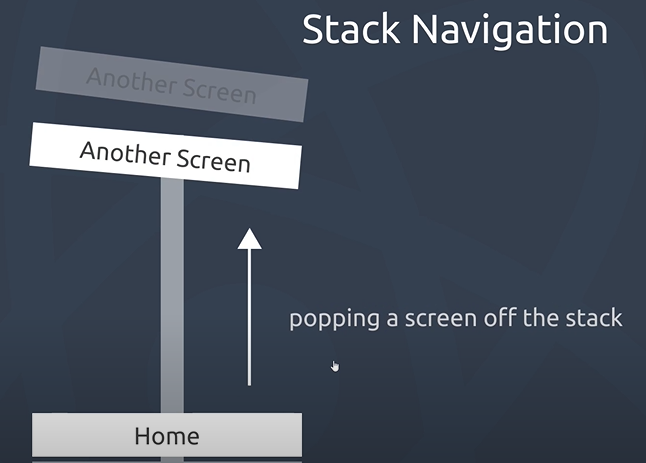
So, we need to install.

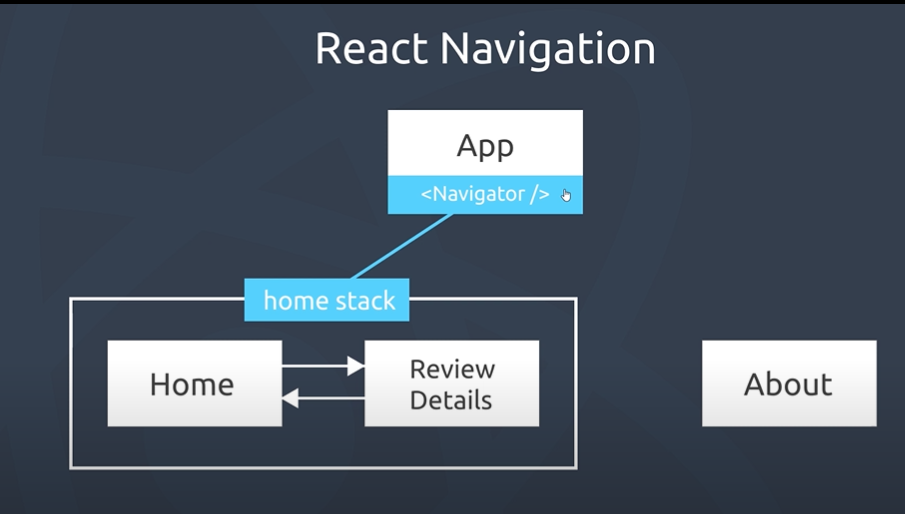
* npm i react-navigation

And install dependencies into an Expo managed project

* npx expo install react-native-screens react-native-safe-area-context
  + 1. Stack Navigator
       1. Definition

Stack Navigator is just like a Pile.





* We need to install (npm i react-navigation-stack)
  + - 1. Navigating Around

App.js

import HomeStack from "./routes/homeStack";

export default function App() {

  return (

  <HomeStack/>

  );

}

homeStack.js

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

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

import Home from "../screens/Home";

import ReviewDetails from "../screens/ReviewDetails";

const Stack = createNativeStackNavigator()

export default function HomeStack() {

  return (

    <NavigationContainer initialRouteName="Home">

    <Stack.Navigator>

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

      <Stack.Screen name="ReviewDetails" component={ReviewDetails} />

    </Stack.Navigator>

  </NavigationContainer>

  );

}

Home.js

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

import { globalStyles } from "../styles/styles";

const Home = ({ navigation }) => {

  const pressHandler = () => {

    navigation.navigate("ReviewDetails");

  };

  return (

    <View style={globalStyles.container}>

      <Text style={globalStyles.titleText}>Home Screen</Text>

      <Button title="go to review dets" onPress={pressHandler} />

    </View>

  );

};

export default Home;

ReviewDetails.js

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

import { globalStyles } from "../styles/styles";

const ReviewDetails = ({navigation}) => {

  const pressHandler=()=>{

      navigation.goBack();

  }

  return (

    <View style={globalStyles.container}>

      <Text>ReviewSetails Screen</Text>

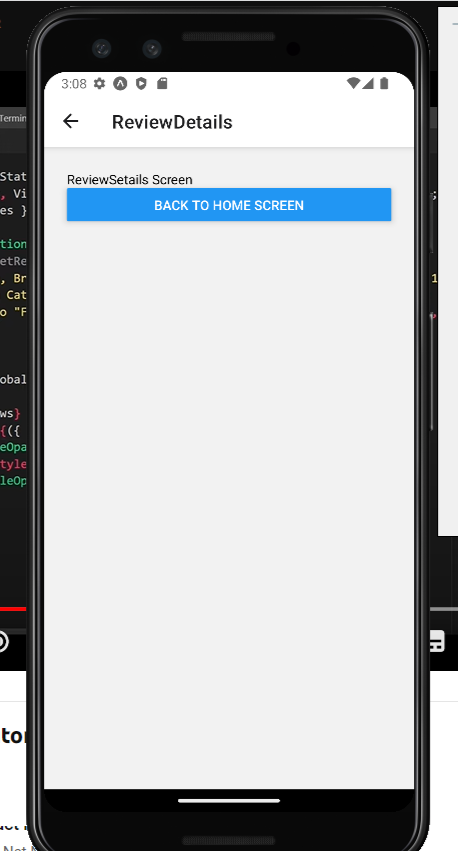
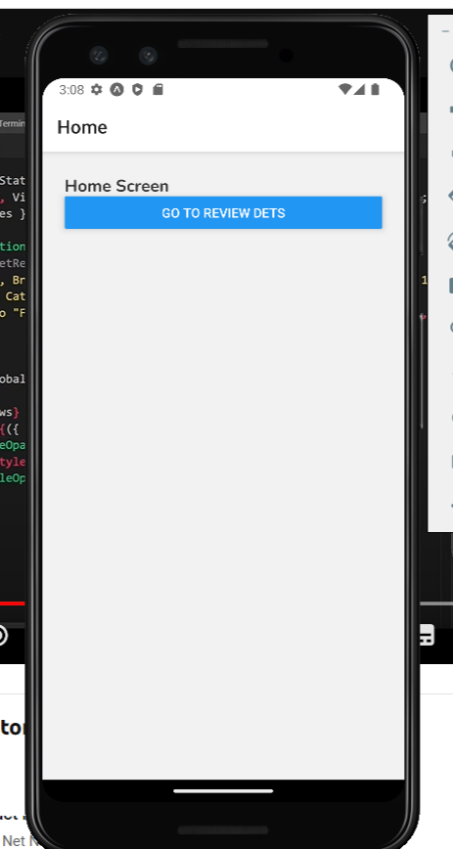
      <Button title='Back to home screen' onPress={pressHandler}/>

    </View>

  );

};

export default ReviewDetails



* + - 1. Passing Data Between Screens
* The second Parameter of navigation.navigate(,) helps us to passing Data it is always an object.
* Pour récupère les donnes on utilise props. route.params.

Home.js

const Home = ({ navigation }) => {

  const [reviews, setReviews] = useState([

    {

      title: "Zelda, Breath of Fresh Air",

      rating: 5,

      body: "lorem ipsum",

      key: "1",

    },

    {

      title: "Gotta Catch Them All (again)",

      rating: 4,

      body: "lorem ipsum",

      key: "2",

    },

    {

      title: 'Not So "Final" Fantasy',

      rating: 3,

      body: "lorem ipsum",

      key: "3",

    },

  ]);

  return (

    <View style={globalStyles.container}>

      <FlatList

        data={reviews}

        renderItem={({ item }) => (

          <TouchableOpacity

            onPress={() => navigation.navigate("ReviewDetails", item)}

          >

            <Text style={globalStyles.titleText}>{item.title}</Text>

          </TouchableOpacity>

        )}

      />

    </View>

  );

};

ReviwDetails.js

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

  const pressHandler = () => {

    navigation.goBack();

  };

  return (

    <View style={globalStyles.container}>

      <Text>{route.params.title}</Text>

      <Text>{route.params.rating}</Text>

      <Text>{route.params.body}</Text>

    </View>

  );

};

* + - 1. Navigation Option

We can make a global Option or an Option for each Route.

* GlobalOption
  + <Stack.Navigator
  + screenOptions={{
  + headerStyle: {
  + backgroundColor: 'red',
  + }, }}>
* Option
  + <Stack.Screen
  + name="Home"
  + component={Home}
  + options={{
  + title: "My home",
  + headerStyle: { backgroundColor: "#eee" },
  + }} />

const Stack = createNativeStackNavigator();

export default function HomeStack() {

  return (

    <NavigationContainer initialRouteName="Home">

      <Stack.Navigator

      screenOptions={{

        headerStyle: {

          backgroundColor: 'red',

        },

      }}>

        <Stack.Screen

          name="Home"

          component={Home}

          options={{

            title: "My home",

            headerStyle: { backgroundColor: "#eee" },

          }}

        />

        <Stack.Screen name="ReviewDetails" component={ReviewDetails} />

      </Stack.Navigator>

    </NavigationContainer>

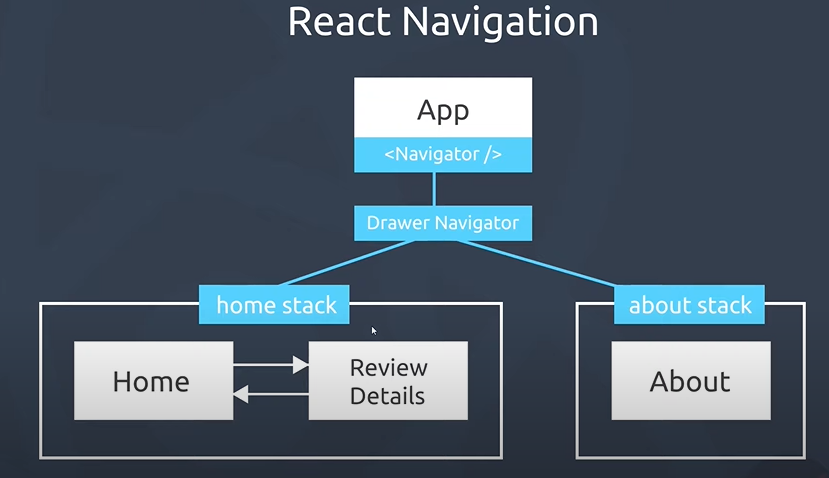
  );

}

* + 1. Draw Navigator
       1. Definition And Instalation And what we will do
* Definition

DrawerNavigator is a type of navigation in React Native and React Navigation library that provides a drawer navigation pattern. It is used to display a navigation drawer on the screen that slides in from the side of the screen. DrawerNavigator allows the user to access navigation links from the side menu, making it a useful addition to the app's navigation structure.

* Installation
  + npm i @react-navigation/drawer
* What we will do



App.js

export default function App() {

  return (

    // <View style={styles.container}>

      <Testdrawer />

    // </View>

  );

}

Drawer.js

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

import { createDrawerNavigator } from "@react-navigation/drawer";

import HomeStack from "./homeStack";

import AboutStack from "./aboutStack";

const Drawer = createDrawerNavigator();

export default function Testdrawer() {

  return (

    <NavigationContainer>

      <Drawer.Navigator initialRouteName="Home">

        <Drawer.Screen name="HomeStack" component={HomeStack} />

        <Drawer.Screen name="AboutStack" component={AboutStack} />

      </Drawer.Navigator>

    </NavigationContainer>

  );

}

homeStck.js

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

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

import Home from "../screens/Home";

import ReviewDetails from "../screens/ReviewDetails";

const Stack = createNativeStackNavigator();

export default function HomeStack() {

  return (

    // <NavigationContainer >

      <Stack.Navigator

      screenOptions={{

        headerStyle: {

          backgroundColor: 'red',

        },

      }}>

        <Stack.Screen

          name="Home"

          component={Home}

          options={{

            title: "My home",

            headerStyle: { backgroundColor: "#eee" },

          }}

        />

        <Stack.Screen name="ReviewDetails" component={ReviewDetails} />

      </Stack.Navigator>

    // </NavigationContainer>

  );

}

AboutStack.js

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

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

import About from "../screens/About";

const Stack = createNativeStackNavigator();

export default function AboutStack() {

  return (

    // <NavigationContainer>

      <Stack.Navigator

      screenOptions={{

        headerStyle: {

          backgroundColor: 'red',

        },

      }}>

        <Stack.Screen

          name="About"

          component={About}

          options={{

            title: "About Page",

            headerStyle: { backgroundColor: "#eee" },

          }}

        />

      </Stack.Navigator>

    // </NavigationContainer>

  );

}

* + - 1. Using Images

We have already the images that we will use in assets/

ReviewDetails.js

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

  const pressHandler = () => {

    navigation.goBack();

  };

  return (

    <View style={globalStyles.container}>

      <Card>

        <Text>{route.params.title}</Text>

        <Text>{route.params.rating}</Text>

        <View style={styles.rating}>

          <Text>GameZone rating: </Text>

          <Image source={images.ratings[route.params.rating]} />

        </View>

      </Card>

    </View>

  );

};

const styles = StyleSheet.create({

  rating: {

    flexDirection: "row",

    justifyContent: "center",

    paddingTop: 16,

    marginTop: 16,

    borderTopWidth: 1,

    borderTopColor: "#eee",

  },

});

Style.js (normalement on doit pas le metre sur style.js car c’est pas u style )

export const images = {

  ratings: {

    '1': require('../assets/rating-1.png'),

    '2': require('../assets/rating-2.png'),

    '3': require('../assets/rating-3.png'),

    '4': require('../assets/rating-4.png'),

    '5': require('../assets/rating-5.png'),

  }

};

* + - 1. Using ImagesBackground

export default function Header({ title, navigation }) {

  const openMenu = () => {

    navigation.openDrawer();

  }

  return (

    <ImageBackground source={require('../assets/game\_bg.png')} style={styles.header}>

      {/\* <MaterialIcons name='menu' size={28} onPress={openMenu} style={styles.icon} /> \*/}

      <View style={styles.headerTitle}>

        <Image source={require('../assets/heart\_logo.png')} style={styles.headerImage} />

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

      </View>

    </ImageBackground>

  );

}

const styles = StyleSheet.create({

  header: {

    width: '100%',

    height: '100%',

    flexDirection: 'row',

    alignItems: 'center',

    justifyContent: 'center',

  },

  headerText: {

    fontWeight: 'bold',

    fontSize: 20,

    color: '#333',

    letterSpacing: 1,

  },

  icon: {

    position: 'absolute',

    left: 16,

  },

  headerTitle: {

    flexDirection: 'row'

  },

  headerImage: {

    width: 26,

    height: 26,

    marginHorizontal: 10

  },

});

* 1. Modal

Modals are used in React Native to display a temporary, full-screen view that covers the current screen. They are used to show additional content that may be relevant to the user, such as a form, a menu, or a confirmation prompt. Modals provide a convenient way to display content that needs to be temporarily visible without navigating to a new screen. They allow users to interact with the content within the modal, and then dismiss it when they are done, all without leaving the current screen.

const Home = ({ navigation }) => {

  const [modalOpen, setModalOpen] = useState(false);

  const [reviews, setReviews] = useState([

    {

      title: "Zeld, Breath of Fresh Air",

      rating: 5,

      body: "lorem ipsum",

      key: "1",

    },

    {

      title: "Gotta Catch Them All (again)",

      rating: 4,

      body: "lorem ipsum",

      key: "2",

    },

    {

      title: 'Not So "Final" Fantasy',

      rating: 3,

      body: "lorem ipsum",

      key: "3",

    },

  ]);

  return (

    <View style={globalStyles.container}>

  <Modal visible={modalOpen} animationType='slide'>

        <View style={styles.modalContent}>

          <MaterialIcons

            name='close'

            size={24}

            style={{...styles.modalToggle, ...styles.modalClose}}

            onPress={() => setModalOpen(false)}

          />

          <Text>Hello from the modal :)</Text>

        </View>

      </Modal>

      <MaterialIcons

        name='add'

        size={24}

        style={styles.modalToggle}

        onPress={() => setModalOpen(true)}

      />

      <FlatList

        data={reviews}

        renderItem={({ item }) => (

          <TouchableOpacity

            onPress={() => navigation.navigate("ReviewDetails", item)}

          >

            <Card>

            <Text style={globalStyles.titleText}>{item.title}</Text>

            </Card>

          </TouchableOpacity>

        )}

      />

    </View>

  );

};

const styles = StyleSheet.create({

  modalToggle: {

    justifyContent: 'center',

    alignItems: 'center',

    marginBottom: 10,

    borderWidth: 1,

    borderColor: '#f2f2f2',

    padding: 10,

    borderRadius: 10,

    alignSelf: 'center',

  },

  modalClose: {

    marginTop: 20,

    marginBottom: 0,

  },

  modalContent: {

    flex: 1,

  }

});

export default Home;

* 1. Formik & Yup
     1. Formik

Formik helps us to create Forms.

* Install
  + npm install formik
* <Formik  **…** >
  + initialValues={{ title: "", body: "", rating: "" }}
    - les Valeur initial
  + onSubmit={(values, actions) => { }
    - Value: return us all the values of our Form
    - Actions:
      * Is an object helps us to us many methods like :
        + actions.resetForm() 🡺 Reset Form helps Us to reset all the value of our form
* <Formik  **…** > {Children}</Formik>
  + In the children Component we can add
    - style={globalStyles.input}
    - placeholder="Review title"
    - onChangeText={props.handleChange("title")}
    - value={props.values.title}

reviewForm.js

import React from "react";

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

import { globalStyles } from "../styles/styles";

import { Formik } from "formik";

export default function ReviewForm({ addReview }) {

  return (

    <View style={globalStyles.container}>

      <Formik

        initialValues={{ title: "", body: "", rating: "" }}

        onSubmit={(values, actions) => {

          actions.resetForm();

          //Reset Form helps Us to reset all the value of our form

          addReview(values);

        }}

      >

        {(props) => (

          <View>

            <TextInput

              style={globalStyles.input}

              placeholder="Review title"

              onChangeText={props.handleChange("title")}

              value={props.values.title}

            />

            <TextInput

              style={globalStyles.input}

              multiline

              placeholder="Review details"

              onChangeText={props.handleChange("body")}

              value={props.values.body}

            />

            <TextInput

              style={globalStyles.input}

              placeholder="Rating (1 - 5)"

              onChangeText={props.handleChange("rating")}

              value={props.values.rating}

              keyboardType="numeric"

            />

            <Button

              color="maroon"

              title="Submit"

              onPress={props.handleSubmit}

            />

          </View>

        )}

      </Formik>

    </View>

  );

}

Home.js

 const addReview = (review) => {

    review.key = Math.random().toString();

    setReviews((currentReviews) => {

      return [review, ...currentReviews];

    });

    setModalOpen(false);

  };

<View style={globalStyles.container}>

      <Modal visible={modalOpen} animationType="slide">

        <View style={styles.modalContent}>

          <MaterialIcons

            name="close"

            size={24}

            style={{ ...styles.modalToggle, ...styles.modalClose }}

            onPress={() => setModalOpen(false)}

          />

          <ReviewForm addReview={addReview} />

        </View>

      </Modal>

      <MaterialIcons

        name="add"

        size={24}

        style={styles.modalToggle}

        onPress={() => setModalOpen(true)}

      />

* + 1. Validation Using Yup

Installation

* npm install yup

Why we use yup

* Yup is a JavaScript library used in React Native to validate and cast data. It provides an easy-to-use, expressive API for defining object validation and data transformation rules. With Yup, you can validate user input, for example, ensuring that a field is required or that it matches a specific format like an email address. This helps you maintain high-quality data in your React Native application and ensure that the input from users meets the requirements of your application.

Formik and Formik Error With Yup

OnBlur (Definition general dans notre code il vas nous aide pour afficher le message d’erreur avant quad click sur Submit on touche un autre input et il va etre afficher ) : “onBlur” event in Formik React is used to trigger validation of a form field when it loses focus. This allows you to validate the field’s value as soon as the user moves away from it, providing immediate feedback to the user. The onBlur event handler can also be used to perform additional actions, such as updating the state of the form or performing data manipulation, after a field’s value has been validated.

Yup.object({}): pour crée notre validate shema

reiewForm.js

import React from "react";

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

import { globalStyles } from "../styles/styles";

import { Formik } from "formik";

import \* as yup from "yup";

const reviewSchema = yup.object({

  title: yup.string().required().min(4),

  // String Obligtoire And minimum of the caracter are obligatoire

  body: yup.string().required().min(8),

  rating: yup

    .string()

    .required()

    .test("is-num-1-5", "Rating must be a number 1 - 5", (val) => {

      return parseInt(val) < 6 && parseInt(val) > 0;

    }),

  // String Obligtoire And value in required to be between 1 and five

  //Test(,,)The first argument is the title of our test the second argument

  //   is for the message function and the third argument is for our condition test

});

export default function ReviewForm({ addReview }) {

  return (

    <View style={globalStyles.container}>

      <Formik

        initialValues={{ title: "", body: "", rating: "" }}

        validationSchema={reviewSchema}

        onSubmit={(values, actions) => {

          actions.resetForm();

          //Reset Form helps Us to reset all the value of our form

          addReview(values);

        }}

      >

        {(props) => (

          <View>

            <TextInput

              style={globalStyles.input}

              placeholder="Review title"

              onChangeText={props.handleChange("title")}

              onBlur={props.handleBlur('title')}

              value={props.values.title}

            />

            {/\* only if the left value is a valid string, will the right value be displayed \*/}

            <Text style={globalStyles.errorText}>

              {props.touched.title && props.errors.title}

{/\* props.touched.title qunad on valide notre Forms mais si on ajoute Onblur c’est quand on touche dans une autre place \*/}

            </Text>

            <TextInput

              style={globalStyles.input}

              multiline

              placeholder="Review details"

              onChangeText={props.handleChange("body")}

              onBlur={props.handleBlur('body')}

              value={props.values.body}

            />

            <Text style={globalStyles.errorText}>

              {props.touched.body && props.errors.body}

            </Text>

            <TextInput

              style={globalStyles.input}

              placeholder="Rating (1 - 5)"

              onChangeText={props.handleChange("rating")}

              onBlur={props.handleBlur('rating')}

              value={props.values.rating}

              keyboardType="numeric"

            />

            <Text style={globalStyles.errorText}>

              {props.touched.rating && props.errors.rating}

            </Text>

            <Button

              color="maroon"

              title="Submit"

              onPress={props.handleSubmit}

            />

          </View>

        )}

      </Formik>

    </View>

  );

}

: