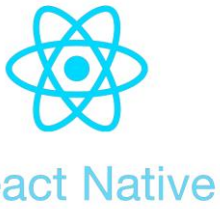


DOCUMENT TECHNIQUE DE LA FORMATION DE REACT NATIVE

Réalisé par : HAFDI AHMED

Encadré par : Mr SERRAJ Mohammed Amine

Table des matières

Table des Figures

Liste des Tableaux :

CHAPITRE 1 : SET UP POUR L'ENVIRONEMMENT WINDOWS

Figure 1:react native logo

React Native vous permet de créer des applications mobiles en utilisant uniquement JavaScript. Il utilise la même conception que React, vous permettant de composer une interface utilisateur mobile riche à partir de composants déclaratifs. Avec React Native, vous ne créez pas une «application Web mobile», une «application HTML5» ou une «application hybride».

Vous construisez une véritable application mobile qui ne se distingue pas d'une application créée avec Objective-C ou Java. React Native utilise les mêmes éléments fondamentaux que les applications iOS et Android classiques. Vous venez de mettre ces blocs de construction ensemble en utilisant JavaScript et React.

Il est open-source et maintenu par Facebook.

• Site Internet (https://reactnative.dev/)

• Documentation (https://reactnative.dev/docs/getting-started)

• GitHub Repository(https://github.com/facebook/react-native)

Configuration pour Windows :

Remarque: vous ne pouvez pas développer d'applications réactives pour iOS sur Windows, mais uniquement des applications Android réactives.

Outils / Environnement

- Windows 10
- outil de ligne de commande (par exemple, ligne de commande Powershell ou Windows)
- Chocolaté (étapes pour configurer via PowerShell via ce lien https://blog.fbalashov.com/2016/07/react-native-android-apps-on-windows.html#setup-choco)
- Le JDK (version 8)
- Studio Android Une machine Intel avec la technologie de virtualisation activée pour HAXM (facultatif, nécessaire uniquement si vous souhaitez utiliser un émulateur)

1) **Configurez votre machine pour réagir au développement natif :**
   Démarrez la ligne de commande en tant qu'administrateur exécutez les commandes suivantes:

   choco install nodejs.install
   choco install python2

   Redémarrez la ligne de commande en tant qu'administrateur pour pouvoir exécuter npm
   npm install -g react-native-cli

2) Définissez vos variables d'environnement :

Ouvrez la fenêtre Variables d'environnement en naviguant vers: [Clic droit] Menu "Démarrer" -> Système -> Paramètres système avancés -> Variables d'environnement Dans la section inférieure, recherchez la variable système "Path" et ajoutez l'emplacement d'installation de react-native à l'étape 1. Si vous n'avez pas ajouté de variable d'environnement ANDROID_HOME, vous devrez également le faire ici. Dans la fenêtre "Variables d'environnement", ajoutez une nouvelle variable système nommée "ANDROID_HOME" et la valeur correspondant au chemin d'accès à votre SDK Android. Redémarrez ensuite la ligne de commande en tant qu'administrateur pour pouvoir y exécuter des commandes réactives.

3) Créez votre projet

Créez votre projet En ligne de commande, accédez au dossier dans lequel vous souhaitez placer votre projet et exécutez la commande suivante:

react-native init ProjectName

4) Lancez votre projet

Démarrez un émulateur depuis Android Studio Accédez au répertoire racine de votre projet en ligne de commande et exécutez-le:

cd ProjectName

react-native run-android

CHAPITRE 2 : BONJOUR LE MONDE

On crée un projet avec la commande :

expo init Hello World

on lance le project avec la commande : (on lance la commande sur le path du projet)

expo start

on ouvrit le fichier App.js

Après on écrit *<Text>Bonjour Tout le Monde </Text> :*

On rafraichit le fichier avec la commande « control + S »

Et voila Félicitations! Vous avez écrit avec succès votre premier Hello World!

Bonjour Tout le monde!

```
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>Bonjour Tout le Monde </Text>
      <StatusBar style="auto" />
    </View>
  );
}

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

```

Figure 2: Hello World

Vous devriez voir Hello World! écrit à l'écran!

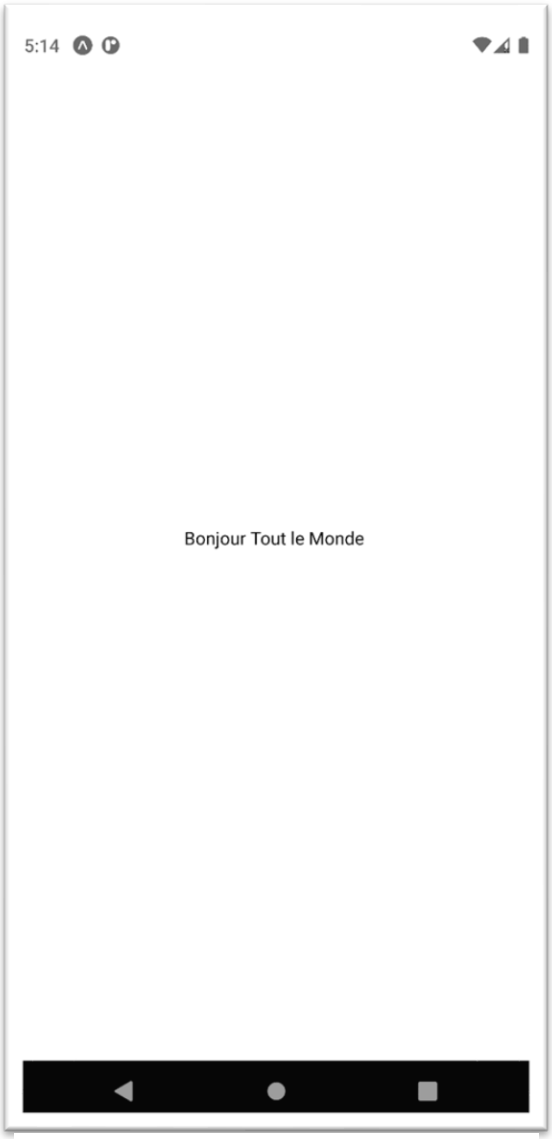


Figure 3: Hello World Screen

Chapitre 3 : Coiffant (stylesheet)

Introduction :

Les styles sont définis dans un objet JSON avec des noms d'attributs de style similaires, comme dans CSS. Un tel objet peut soit être mis en ligne dans le style prop d'un composant, soit être transmis à la fonction StyleSheet.create(StyleObject) et être stocké dans une variable pour un accès en ligne plus court en utilisant un nom de sélecteur similaire à une classe. en CSS.

Syntaxe :

- <Component style={styleFormaStyleSheet }/>
- <Component style={styleObject }/>
- <Component style={style1,style2 }/>

Remarques :

La plupart des styles React Native sont leurs formulaires CSS, mais dans un cas camel. Ainsi, la text-decoration devient textDecoration . Contrairement aux CSS, les styles ne sont pas hérités. Si vous souhaitez que les composants enfants héritent d'un certain style, vous devez le fournir explicitement à l'enfant. Cela signifie que vous ne pouvez pas définir une famille de polices pour une View entière. La seule exception à cette règle est le composant Text : les Text imbriqués héritent de leurs styles parents.

Examples :

Il y a plusieurs manière pour définir un style :

- Style en ligne :
  <Text style={{color : 'red}} >Red Text </Text>

- Styling à l'aide d'une feuille de style :

```
Const styles = StyleSheet.create {{

        red : {
                color :  'red'

        },

        }}
```

- Styling à l'aide de plusieurs styles :

```
const styles = StyleSheet.create({
red: { color: 'red' },
greenUnderline: { color: 'green',
textDecoration: 'underline' },
```

```
big: { fontSize: 30 } });

//code of the class
<Text  style={{styles.red,styles.big  > Big red  </Text>

<Text  style={{styles.red,styles.greenUnderline > Big red  </Text>
```

Chapitre 6 : Composants

Examples :

Composant de base :

```
import React, { Component } from 'react'
import { View, Text, AppRegistry } from 'react-native'
class Example extends Component {
 render () {
 return (
 <View>
 <Text> I'm a basic Component </Text>
 </View>
 )
 }
}
AppRegistry.registerComponent('Example', () => Example)
```

Figure 4: basic component

Composant avec état
Ces composants auront des états changeants.(componant with states)

On utiliser un constructor pour créer un état / state car on utilise ici une Classe comme un composant

```
import React, { Component } from 'react'
import { View, Text, AppRegistry } from 'react-native'
class Example extends Component {
  Constructor (props){
    super(props)

    this.state={
      name:"state 1"
    }

  }
 render () {
 return (
 <View>
 <Text> I'm a basic Component </Text>
 </View>
 )
 }
}
AppRegistry.registerComponent('Example', () => Example)
```

Figure 5: component avec état

Composant sans état (composant without state)

(ce sont les fonctions , funtional componants are stateless , class componants are statefull)

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

function App(props) {
  return (
    <View style={styles.container}>
      <Text style={{color:'black'}} >composant sans état</Text>


    </View>
  );
}

export default App;
const styles = StyleSheet.create({
  container  : {
    flex : 1,
    justifyContent:'center',
    alignItems:'center'

  },
})
```

Figure 6: composant sans état

Chapitre 7 : Disposition

Examples :

Flexbox :

Flexbox est une mode de mise en page permettant la disposition des éléments sur une page de manière à ce que les éléments se comportent de manière prévisible lorsque la mise en page doit

Prendre en charge différents tailles d'écran et différents périphériques d'affichage .par défaut ,

Flexbox représente les enfants (childreen) sous forme d'une colonne (column), Mais on peut le changer en utilisant « flexDirection : 'row » dans le styling .

flexDirection

```
import React from 'react';
import { StyleSheet, Text, View } from 'react-native';
function App(props) {
  return (
    <View style={styles.container}>
      <Text style={{color:'black'}} >flexDirection : row</Text>
      <View style={styles.boxes_container}>
      <View style={styles.box} ></View>
      <View style={styles.box} ></View>
      <View style={styles.box} ></View>
      </View>
    </View>
  );
}
export default App;
const styles = StyleSheet.create({
  container  : {
    flex : 1,justifyContent:'center',alignItems:'center',
  },
  boxes_container:{
    flexDirection:'row'
  },
  box : {
    margin:10,backgroundColor:'green', height:50,width:50,
  },
})
```

Figure 7: flexdirection : 'row'

screen :

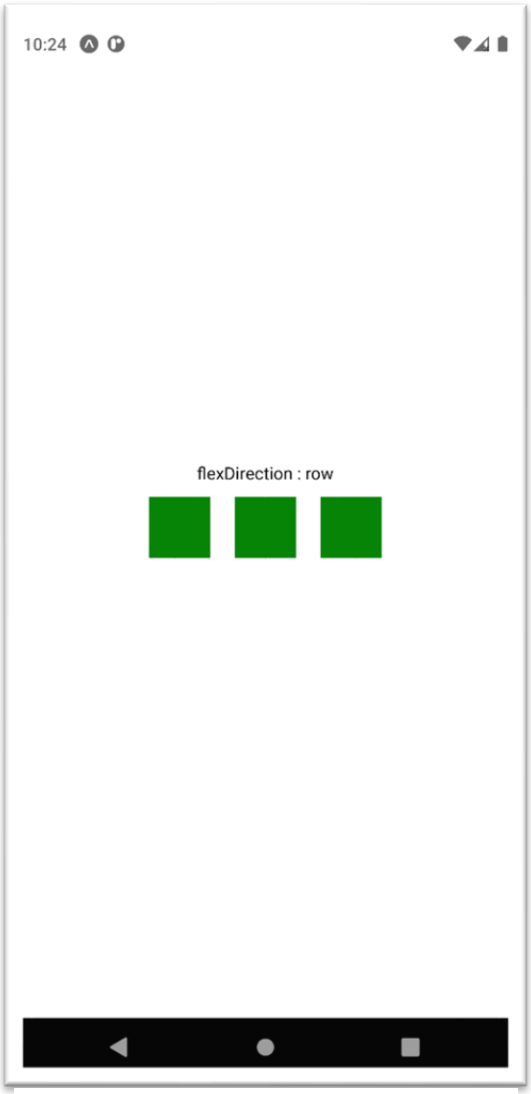


Figure 8 : flexdirection row screen

Ila ya plusieurs possibilité pour changer vers une colonne inversé ou ligne inversé (flexDirection : ‘row-reverse ‘ or ‘column-reverse ‘)

Chaptire 8 : States (les états)

Syntaxe :

- Void **setState** (fonction |object nextState , [rappel de fonction])

Examples :

Pour créer des applications qui contient par exemple des composants qui ont besoin d'être changé instantanément et étre affiché sur l'écran , alors React Native offre les états ou states ,

Pour modifer la vue d'une application , on peut utiliser **setState** ,setState effectue une fusion superficielle entre l'état nouveau et précédent et déclaneche un re-render du composant .

Pour bien expliquer setState et comment ça fonctionne je vais prendre l'exemple d'un compteur avec button

SetState :

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

function States(props) {
    const [counter,setCounter]=useState(1)

    return (
        <View style={styles.Counter_Container}>
                <Text>Counter Value : {counter}</Text>
                <Button  title="add" onPress={()=>setCounter(counter+1)}/>
        </View>
    );
}
export default States;
const styles = StyleSheet.create({
    Counter_Container: {
        flex :1,
        justifyContent:'center',
        alignItems:'center',
    },

})
```

Figure 9: counter add

Screen :

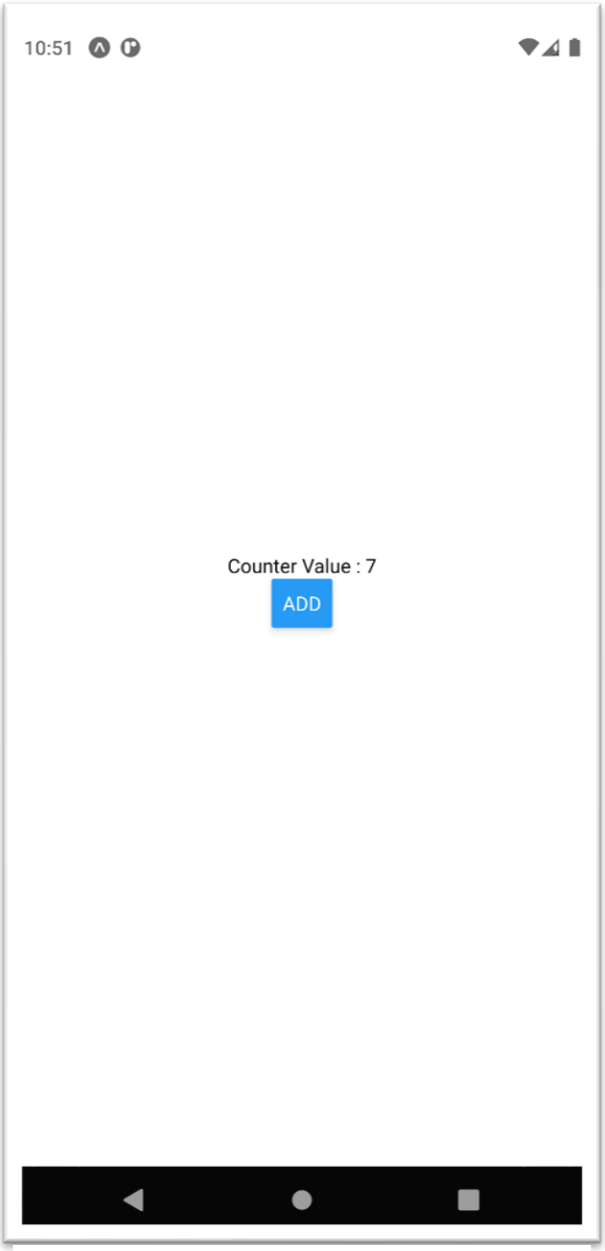


Figure 10: counter add screen

Chapitre 10 : Exécuter une application sur l'appareil (version android) & Expo

Exécuter sur l'appareil :

1. Adb devices (cette commande pour afficher tous les appareil /emulator connectés à l'ordinateur

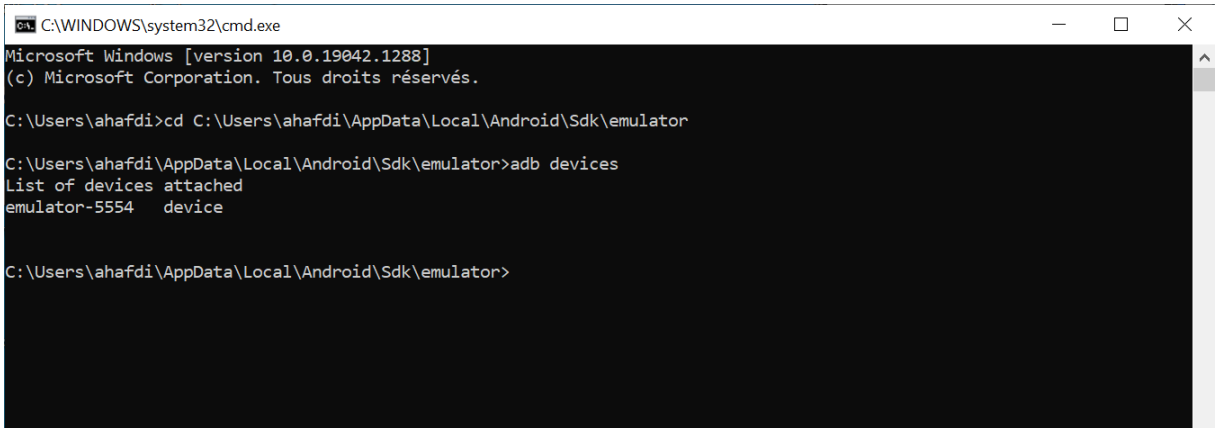


Figure 11: adb devices

2. emulator -avd Pixel_XL_API_30 (lancer l'emulator /simulateur)

3. react-native run-android (exécuter le projet sur le simulateur android)

4. pour Expo (j'utilise expo sur cette documentation)

a. expo start (on lance cette commande sur la destination du projet pour lancer le serveur d'Expo)

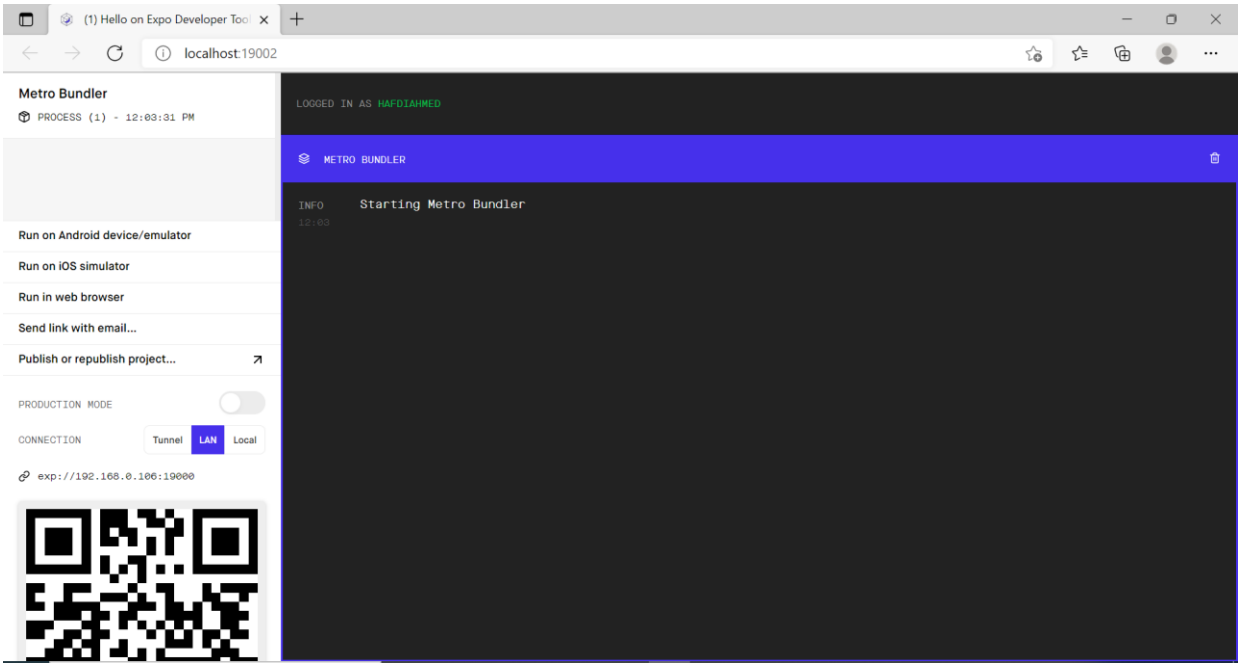


Figure 12: expo start server

b. On clique sur ' run on Android device/emulator' :

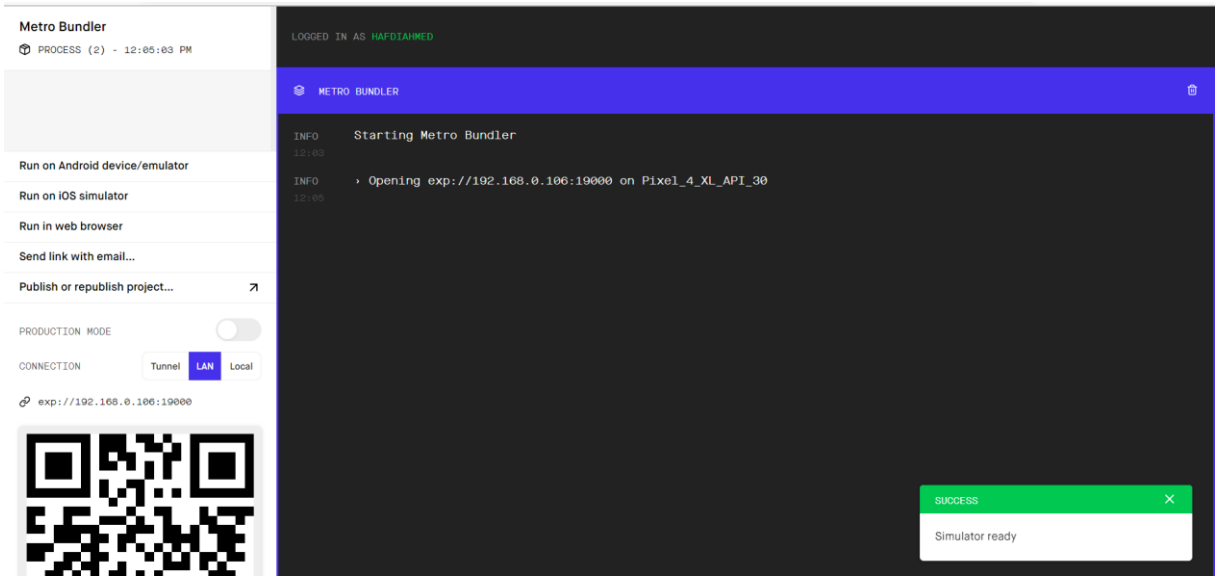


Figure 13: expo run on android emulator

Chapitre 11 : Images

Examples :

Module d'image

On peut importer Image à partir du package « react-native »

- Image locale et Image externe :

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

function Images(props) {
    return (
        <View style={styles.Images_Container} >
            <Text>image locale</Text>
            <Image source={require("../Hello/assets/gear9_1.jpg")} style={styles.image_style}/>
            <Text>image externe</Text>
            <Image   source={{uri : "https://yt3.ggpht.com/ytc/AKedOLQvLGRnf5cnRwTcsXTwRhyI_uwbJ4R_b7HYKm6ciA=s176-c-k-c0x00ffffff-no-rj"}}
            style={styles.image_style}/>
        </View>
    );
}

export default Images;
const styles = StyleSheet.create({
    Images_Container: {flex :1,justifyContent:'center',alignItems:'center',},
    image_style : { height:200,width:200},

})
```

Figure 14: image locale et externe

- **Screen :**

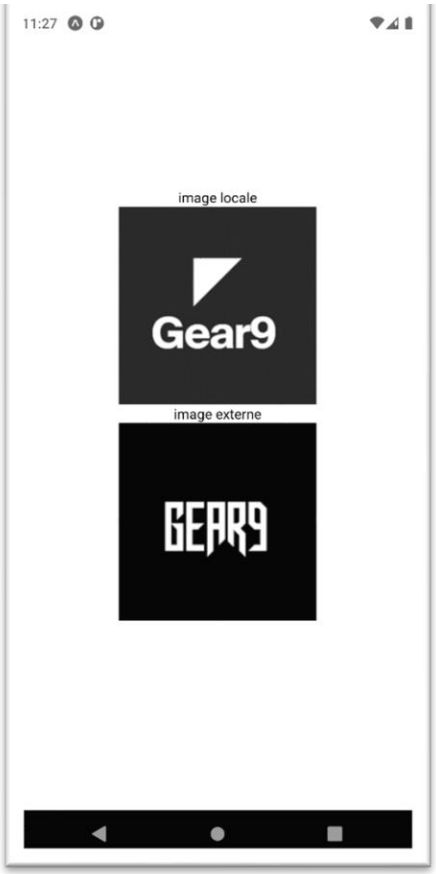


Figure 15: image locale & externe screen

Source d'image conditionnelle :

```
import React from 'react';
import { Image, StyleSheet, Text, View } from 'react-native';
function Images(props) {
    const path_1= require("../Hello/assets/favicon.png");
    const path_2=require("../Hello/assets/gear9_1.jpg");
    let image_path= image_path ? path_2 : path_1 ;
    return (
        <View style={styles.Images_Container} >
            <Text style={styles.text_style}>image conditionel</Text>
            <Image  style={styles.image_style} source={image_path}/>
        </View>
    );
}
export default Images;
const styles = StyleSheet.create({
    Images_Container: {flex :1,justifyContent:'center',alignItems:'center',backgroundColor:"#212120"},
    image_style : { height:150,width:150},
    text_style :{color:"white" ,fontSize:20,},

})
```

Figure 16: image conditionnelle

Screen :

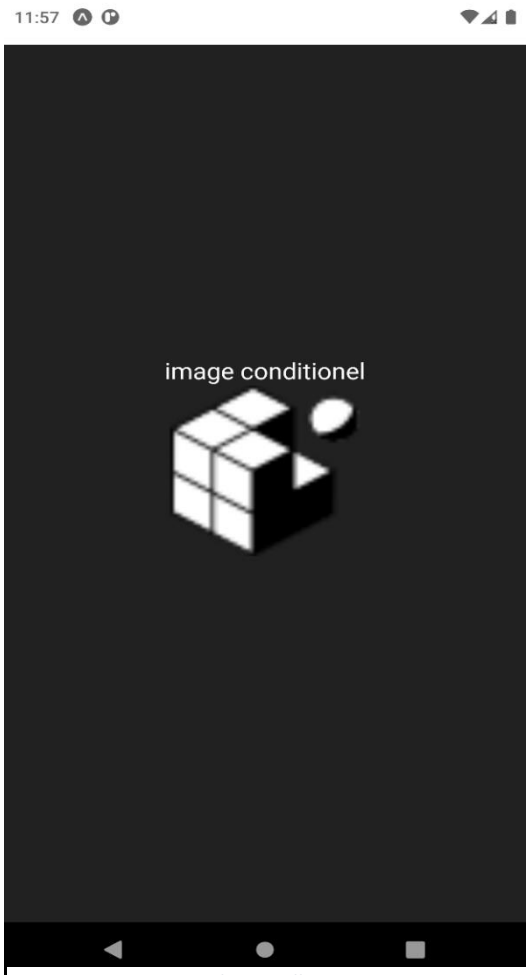


Figure 17: image conditionnelle screen

Chapitre 12 : Instructions en ligne de commande

Examples :

Vérifier la version

- **$ react-native -v**

Examples de sortie

```
C:\Users\ahafdi>react-native -v
react-native-cli: 2.0.1
react-native: n/a - not inside a React Native project directory
```

Figure 18: react native check version

Pour Initialiser

- **$ react-native init MyFirstProjectRN**
- **$ expo init MyFirstProjectEXPO**

Pour courir pour Android

- **$ cd MyFirstProjectRN**
- **$ react-native run-android**

Chapitre 13 : Le débogage

Syntaxe :

- Débogueur

Le débogage avec Expo (Exponent) et VSCode :

- C'est quoi Exponent ?

Figure 19: expo logo

Exponent /Expo est un framework et une plateforme pour les applications de React . c'est un ensemble d'outils et services construits autour de React Native et des plate-formes natives qui aident à développer , créer , déployer et itérer rapidement sur ios ,android et Web à partir de la même base de code JavaScript/TypeScript .

Etapes de débogage par Expo & VScode :

1. Installation de l'extension « React Native Tools » sur VSCode

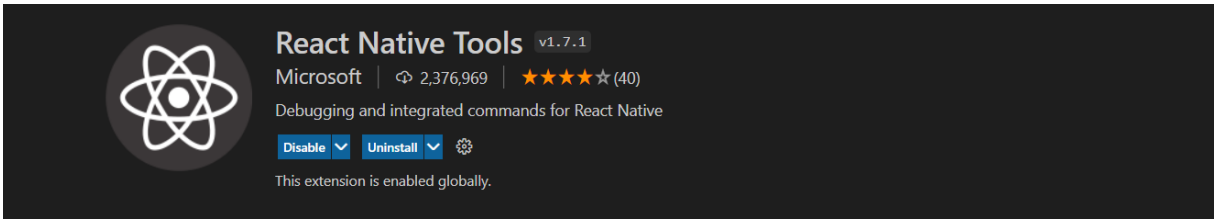


Figure 20: react native tools

2. Après l'ajout de la configuration « Exponent » sur le fichier launche.json dand le dossier. Vscode

```
"configurations": [
    {
        "name": "Debug in Exponent",
        "request": "launch",
        "type": "reactnative",
        "cwd": "${workspaceFolder}",
        "platform": "exponent",
        "expoHostType": "local"
    },
```

Figure 21 : debug in exponent

3. On lance le debug de l'application :

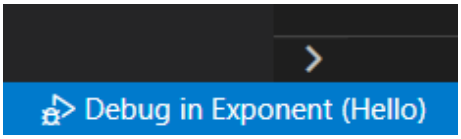


Figure 22: debug

Chapitre 14 : Le Routage

Introduction :

Le routage ou la navigation permet de naviguer entre différents écrans . le routage est très important pour les applications mobile car il fournit un contexte à l'utilisateur sur l'endroit ou il se trouve , il se déplace entre les écrans et le fenêtres .

Exemples :

StepUp Préparation :

Installation des modules :

- npm install @react-navigation/native @react-navigation/native-stack
- expo install react-native-screens react-native-safe-area-context
  (c'est mieux d'utiliser **yarn** cli)

Stack Composant :

Exemple de navigation entre deux screens :

```
import React from 'react';
import { NavigationContainer } from '@react-navigation/native';
import { createStackNavigator } from '@react-navigation/stack';
import { useNavigation } from "@react-navigation/native";
import { Button ,View,Text} from 'react-native';
    const Stack=createStackNavigator()
function Stacks(props) {
    const navigation =useNavigation()
    const First_Screen =()=>{
        return (
            <View>
                <Text>First Screen</Text>
                <Button title="go to 2 screen" onPress={()=>navigation.navigate("Second")} />
            </View>
        );
    }
    const Second_Screen =()=>{
        return (
            <View>
                <Text>Second Screen</Text>
            </View>
        );
    }
    const MyStack =()=>{
        return(
            <Stack.Navigator>
                <Stack.Screen  name ="First" component={First_Screen}/>
                <Stack.Screen name="Second" component={Second_Screen}/>
            </Stack.Navigator>
        );
    }
    return (
            <MyStack/>
    );
}
export default Stacks;
```

Figure 24 : MyStack

Screens :

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

Figure 23 : Navigator

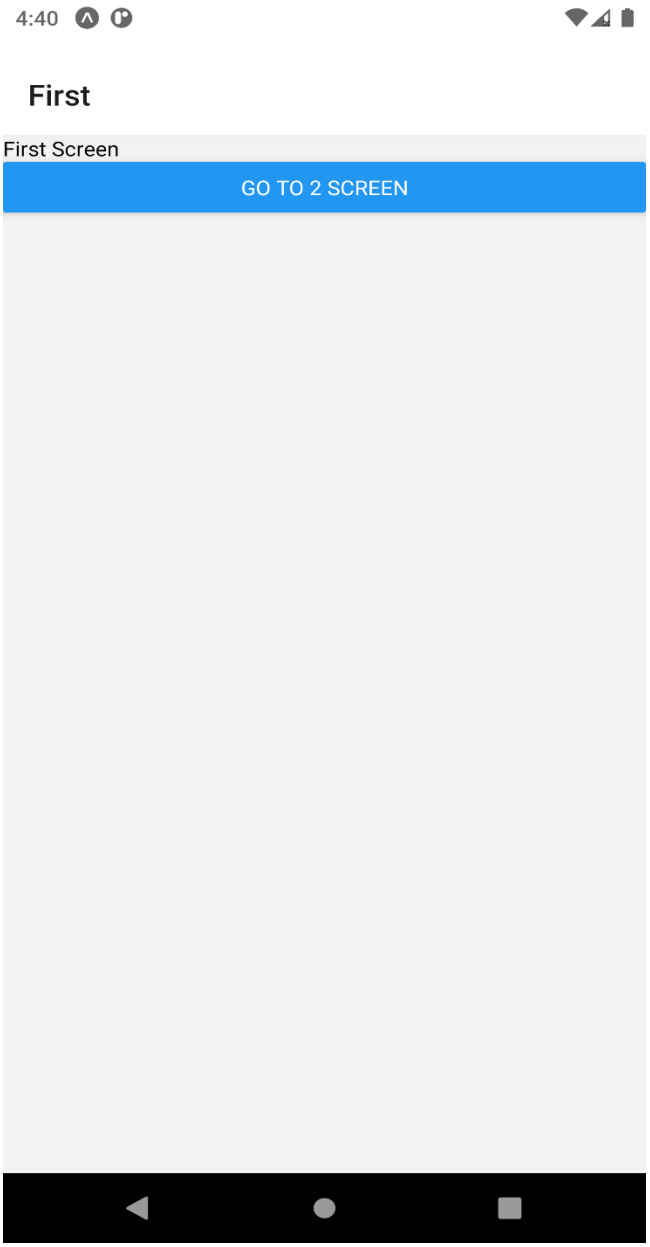


Figure 25: First Screen

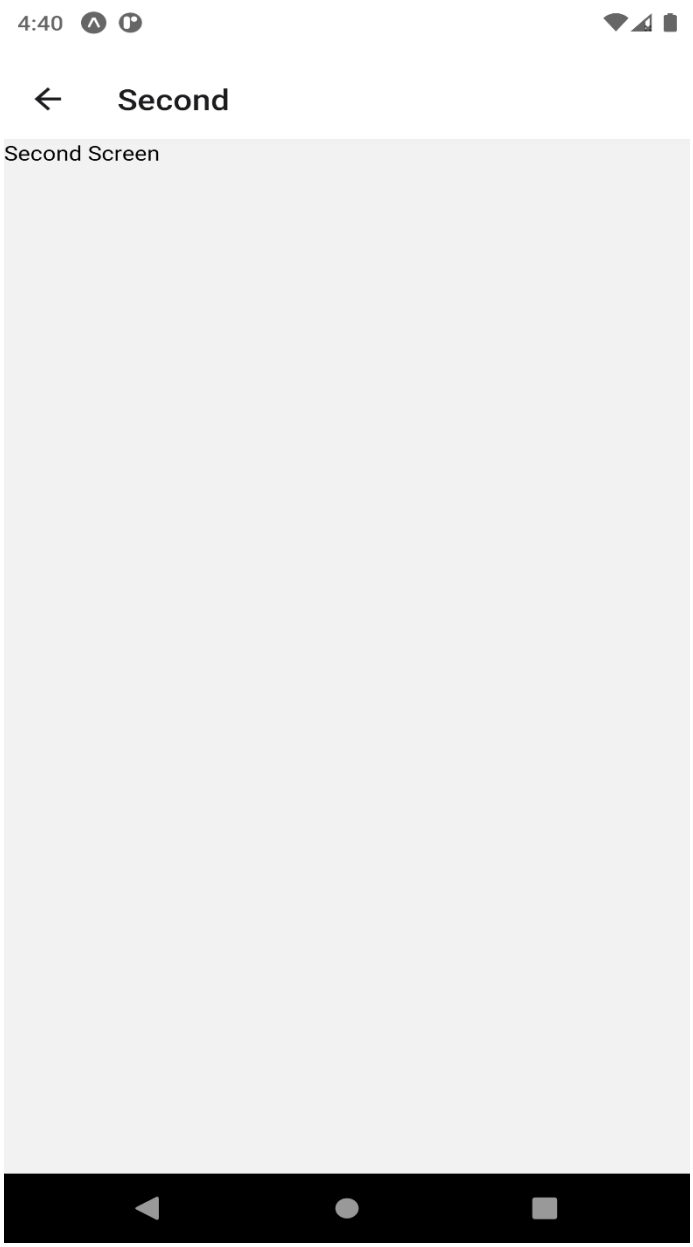


Figure 26: Second Screen

Top Tabs :

```
const Tab = createMaterialTopTabNavigator();
const First_Top=()=>{
    return (
        <View style={styles.Screen}>
            <Text>First Top</Text>
        </View>
    );
}
const Second_Top=()=>{
    return (
        <View style={styles.Screen}>
            <Text>Second Top</Text>
        </View>
   );
}
const MyTops=()=>{
    return(
            <NavigationContainer>
                <Tab.Navigator style={{ marginTop:40,}} >
                <Tab.Screen  name="First Top" component={First_Top}/>
                <Tab.Screen  name="Second Top" component={Second_Top}/>
            </Tab.Navigator>
            </NavigationContainer>
    );
}
function TabTop(props) {
    return (
       <MyTops/>
    );
}
export default TabTop;
const styles = StyleSheet.create({
    Screen: {
            flex : 1,alignItems:'center',justifyContent:'center',
    },
})
```

Figure 27: Top Tabs

Screen :

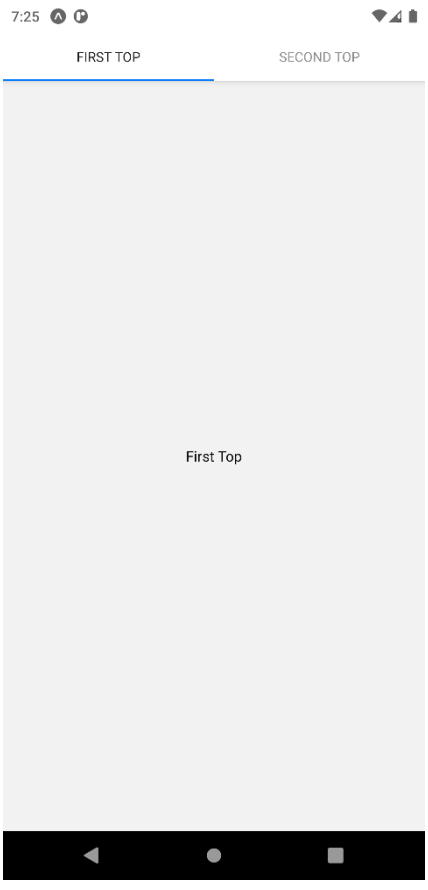


Figure 28: Tab Tops screen

Bottom Tabs

```
function BottomTab(props) {
    const Bottom=createBottomTabNavigator()
    const Tab1 =()=>{
        return (
            <View style={styles.screen}>
                <Text>Tab 1 </Text>
                </View>
        );
    }
    const Tab2 =()=>{
        return (
            <View style={styles.screen} >
            <Text>Tab 2 </Text>
            </View>
        );
    }
    return (
        <NavigationContainer>
            <Bottom.Navigator>
                <Bottom.Screen name="Tab 1" component={Tab1} />
                <Bottom.Screen name ="Tab 2" component={Tab2}/>
            </Bottom.Navigator>
        </NavigationContainer>
    );
}
export default BottomTab;
 const styles = StyleSheet.create({
     screen : { flex :1 , justifyContent:'center',alignItems:"center"}
 })
```

Figure 29: Bottom tabs

Screen :

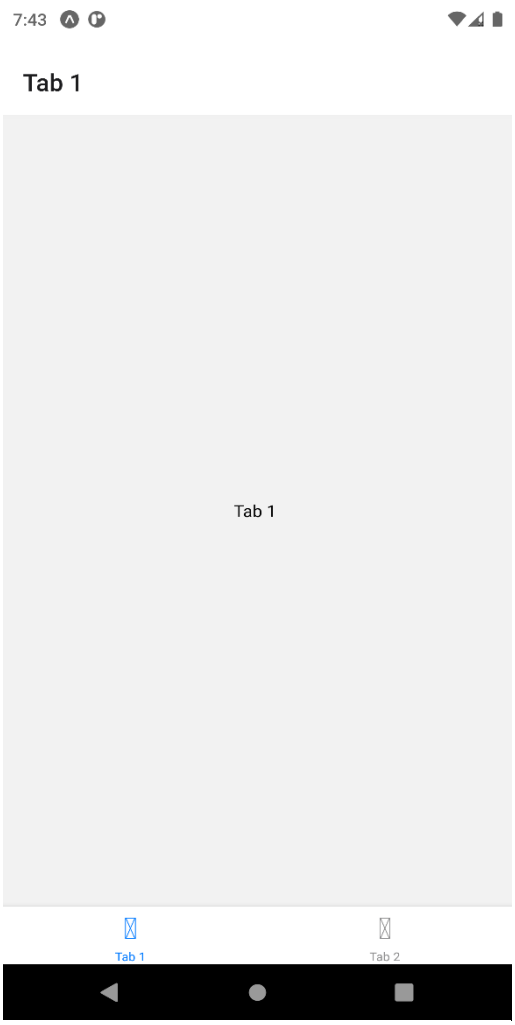


Figure 30: Bottom tabs1

Chapitre 15 : Liaison de l'API native

Introduction :

L'api de liaison permet d'envoyer et de recevoir des liens entre les applications . par exemple ouvrir l'application téléphone avec le numéro composé ou Google Maps .on peut également utiliser **Linking**

Pour répondre aux liens externes depuis d'autres applications .

Pour utiliser 'Linking ' on doit l'importer depuis react-native.

Exemples :

Liens sortants

```
<View style={styles.open_row}>
<Text> call Gear9  </Text>
 <Button  title="call" onPress={()=>Linking.openURL("tel:0666666666")}/>
</View>
```

Figure 31: lien sortant

Liens entrants

```
    <View style={styles.open_row}>
    <Text> open gear9 web site  </Text>
    <Button  title="open website" onPress={()=>Linking.openURL("https://gear9.ma/")}/>
    </View>
```

Figure 32 : lien entrant

Screens :

Figure 34: Linking

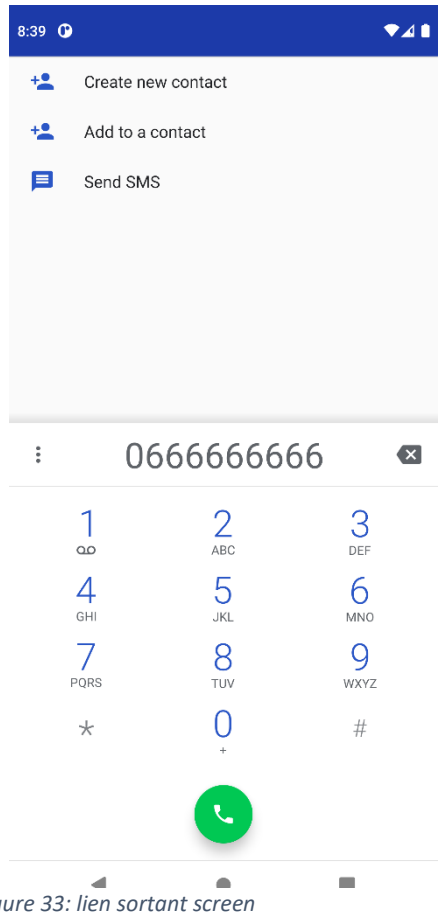


Figure 33: lien sortant screen

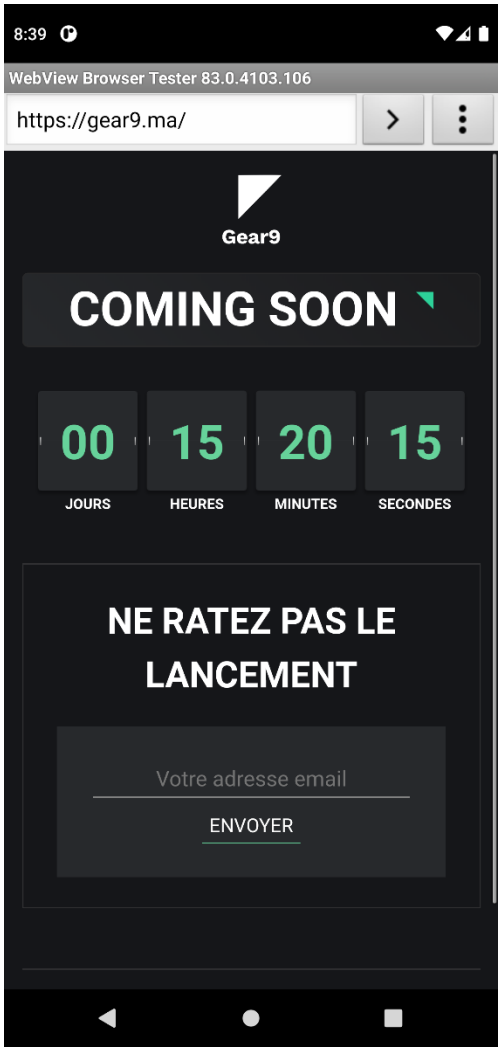


Figure 35: lien entrant

Chapitre 16 : ListView

Exemples :

ListView est un composant cançu pour un affichage efficace des listes de données changeanyes défilant verticalement .il y a plusieurs de types pour l'utilisation de listview FlatList

```
import React from 'react';
import { FlatList, StyleSheet, Text, View } from 'react-native';
const users=[{id : 1,name :"user 1",job :"job 1"},{id : 2,name :"user 2",job :"job 2"},{id : 3,name :"user 3",job :"job 3"},{id : 4,name :"user 4",
function Flatlist_View(props) {
    return (
        <View style={styles.container}>
            <FlatList
           data={users}
            renderItem={({item})=>
        <View style={styles.user}>
            <Text style={styles.Items}>id : {item.id}</Text>
            <Text style={styles.Items}>name: {item.name}</Text>
            <Text style={styles.Items}>job :{item.job}</Text>
        </View>
        }
            />

        </View>
    );
}

export default Flatlist_View;
```

Figure 36:flatList

Screen :

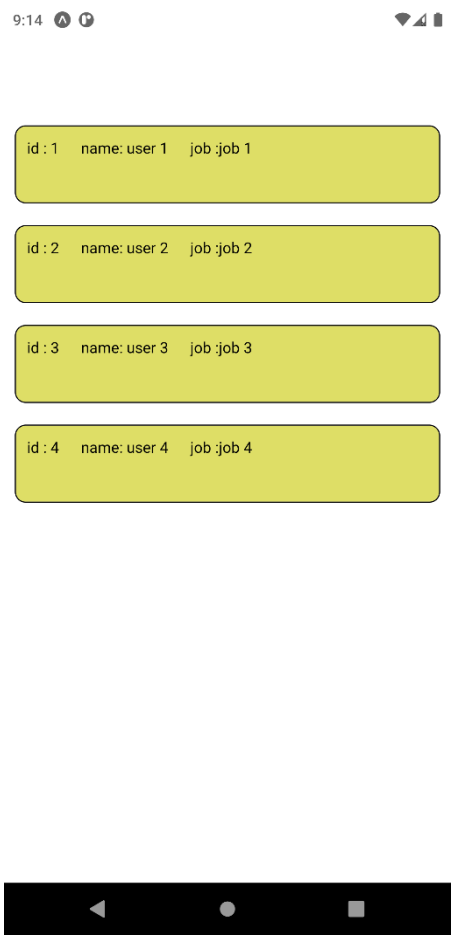


Figure 37: flatlist users

SectionList

```
import React from 'react';
import { SectionList, StyleSheet, Text, View } from 'react-native';
const users=[
        {name : "user 1",data: ["ahmed","samir","jhon"]},
        {name : "user 2",data: ["user 1","samir","jhon"]},
        {name : "user 3",data: ["user 1","samir","jhon"]},
        {name : "user 3",data: ["user 1","samir","jhon"]},]

function Sectionlist_View(props) {
    return (
      <View style={styles.container}>
        <Text style={{fontSize:20,marginHorizontal:60,}}> Users and their friends</Text>
         <SectionList
          sections={users}
         renderItem={({item})=><Text>{item}</Text>}
         renderSectionHeader={({section})=><Text style={styles.name_style}>{section.name}</Text>}
       keyExtractor={(item,index)=>index}
          />
      </View>
    );
}
```

Figure 38:SectionList

Screen :

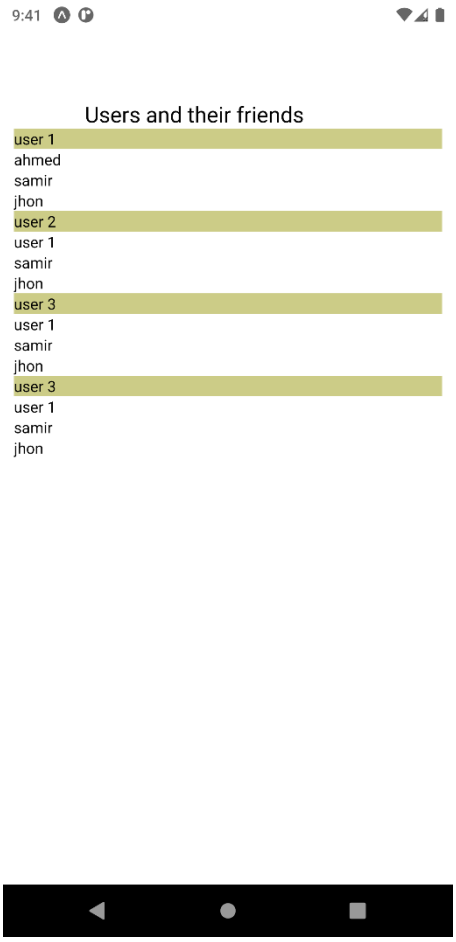


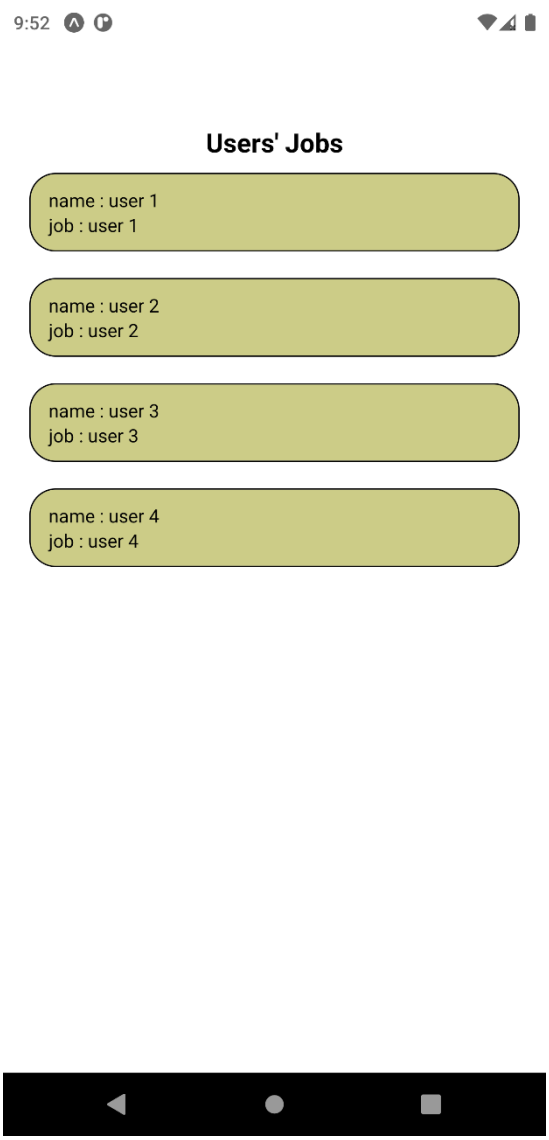
Figure 39: SectionList Screen

Map Array

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

function Map_Array_View(props) {
    const users=[{id : 1,name :"user 1",job :"job 1"},{id : 2,name :"user 2",job :"job 2"},{id : 3,name :"user 3",job :"job 3"},{id : 4,na
    return (
        <View style={styles.container}>
          <Text style={styles.Title_Text}>Users' Jobs</Text>
          {
              users.map((user,index)=>{
                  return (
                      <View style={styles.user_view}>
                      <Text > name : {user.name}</Text>
                      <Text > job : {user.name}</Text>
                      </View>
                  );
              })
          }
        </View>
    );
}

export default Map_Array_View;
```

Figure 40:map array view

screen :



Chapitre 17 : Input Data

Exemples :

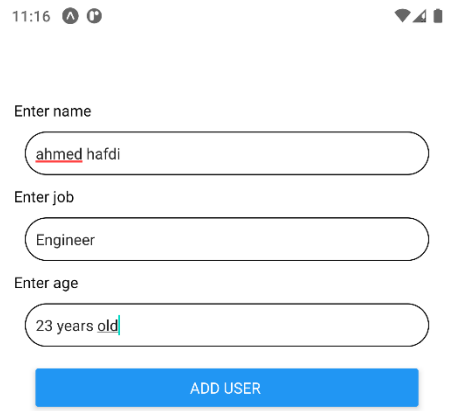
TextInput :

TextInput est un composant pour entrer les données via le clavier .

```
<View style={styles.container}>
    <Text>Enter name</Text>
    <TextInput style={styles.input}
    onChangeText={(text)=>{user.user_name=text}}
    placeholder={"name"}
    />
    <Text>Enter job</Text>
     <TextInput style={styles.input}
    placeholder="job"
    onChangeText={(text)=>{user.user_job=text}}
    />
    <Text>Enter age</Text>
     <TextInput style={styles.input}
     onChangeText={(text)=>{user.user_age=text}}
    placeholder={"age"}
    />
    <View style={styles.button}>
        <Button  title="add user "  onPress={()=>setUser({
            user_name:user.user_name, user_age: user.user_age,user_job:user.user_job
        })}/>
    </View>
</View>
```

Figure 41: TextInput

Screen :



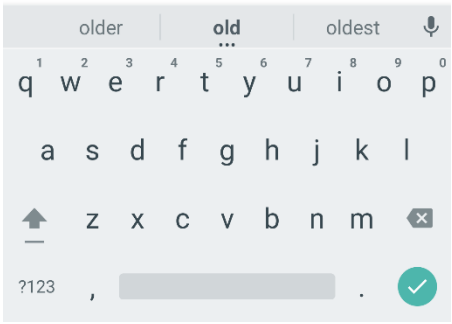


Figure 42: add user screen

Chapitre 18 : Modal

Introduction :

Modal est composant ou un moyen simple de présenter du contenu au-dessus d'une vue englobante.

Paramètres :

Tableau 1: Modal parameters

Fonction	Détails
animationType	('none','slide','fade') il contrôle l'animation modele.
Visible	Est un booléen qui contrôle la visibilité.
onShow	Il permet de passer une fonction qui sera une fois le modal affiché.
transparent	Bool pour la transparence.
onRequestClose (android)	Méthode pour le bouton retourner
onOrientationChange(ios)	Méthode lorsque l'orientation change
SupportedOrientations(IOS)	enum («portrait», «portrait à l'envers», «paysage», «paysage à gauche», «paysage à droite»)

```
import React, { useEffect, useState } from 'react';
import { Button, ImageBackground, Modal, SafeAreaView, SectionList, StyleSheet, Text, TextInput, View } from 'react-native';

function Modal_View_Comp(props) {
    const [open,setopen]=useState(false)
    return (
        <View style={styles.container}>
            <Modal   visible={open}>
                <ImageBackground  style={styles.modal} source={require("../../Project_Hello/assets/gear9_1.jpg")} >
                    <View>
                <Button title="close gear9 logo" onPress={()=>setopen(false)} />
                </View>

                </ImageBackground>

            </Modal>
            <Text>gear 9 logo : </Text>
            <Button title="see gear9 logo" onPress={()=>setopen(true)} />
        </View>
    );
}
```

Figure 43: Modal

Screens :

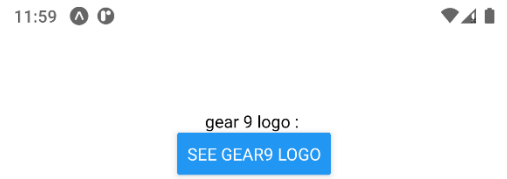


Figure 44: Modal 1

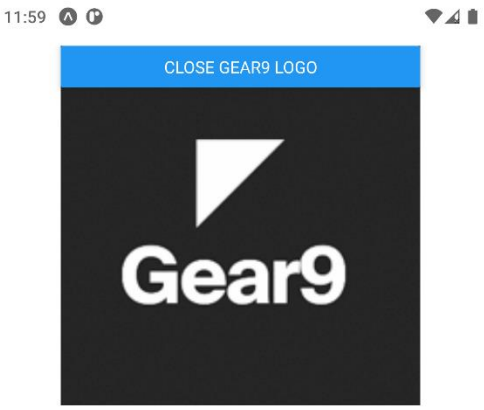


Figure 45: modal 2

Chapitre 19 : Module Plateforme

Introduction :

Module plateforme permet de détecter le type de système d'exploitation.

Exemple :

```
import { Button, StyleSheet, Platform,Text, View, Image } from 'react-native';

function Plateforme_android_ios_comp(props) {
    const android_image="../../Project_Hello/assets/android.png"
    const ios_image="../../Project_Hello/assets/ios.png"
   const Android =()=>{
       return (
<View style={styles.container}>
            <Text style={styles.text}> I am  {Platform.OS}</Text>
            <Image source={require(android_image)} style={styles.image_style}/>
        </View>
       );
   }
   const Ios =()=>{
    return (
<View style={styles.container}>
         <Text style={styles.text}> I am  {Platform.OS}</Text>
         <Image source={require(ios_image)} style={styles.image_style}/>
     </View>
    );
    }
    const Platforme_Ios_or_android =()=>{
       if (Platform.OS==='android')
       {
           return <Android/>
       }
       else {
           return <Ios/>
       }
}
    return (
       < Platforme_Ios_or_android/>
    );
```

Figure 46: Platforme

Scren :

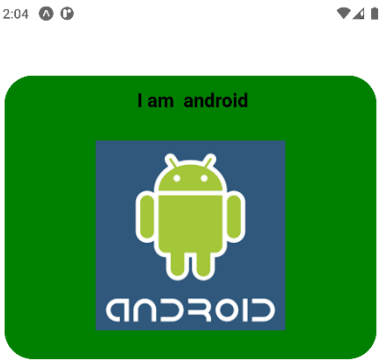


Figure 47: Platform Screen

Chapitre 20 : Notification push

On utilise Push Notification pour réagir avec une application native en utilisant le module react-native-push-notification pour cette documentation j'ai utilisé un autre module pour Expo

C'est expo-permissions et expo-notifications.

Exemple :

```
function PushNotif(props) {
    const notificationListener = useRef();
    const responseListener = useRef();
    useEffect(() => {
        registerForPushNotification().then(token=>console.log(token));
        return () => {
          Notifications.removeNotificationSubscription(notificationListener.current);
          Notifications.removeNotificationSubscription(responseListener.current);    }
      }, [])
     async function registerForPushNotification(){
        const {status} = await Permissions.getAsync(Permissions.NOTIFICATIONS);
        if (status != 'granted') {
          const { status } = await Permissions.askAsync(Permissions.NOTIFICATIONS);
           finalStatus = status;
        }
        if (status !== 'granted') {
          alert('Failed to get push token for push notification!');
          return;
        }
        const token = (await Notifications.getExpoPushTokenAsync()).data;
        return token
      }
    return (
        <View style={styles.container}>
            <Text style={styles.text}> Push Notification RN  expo</Text>
        </View>
    );
```

Figure 48: Push Notification

Expo Push Notifications Tool :

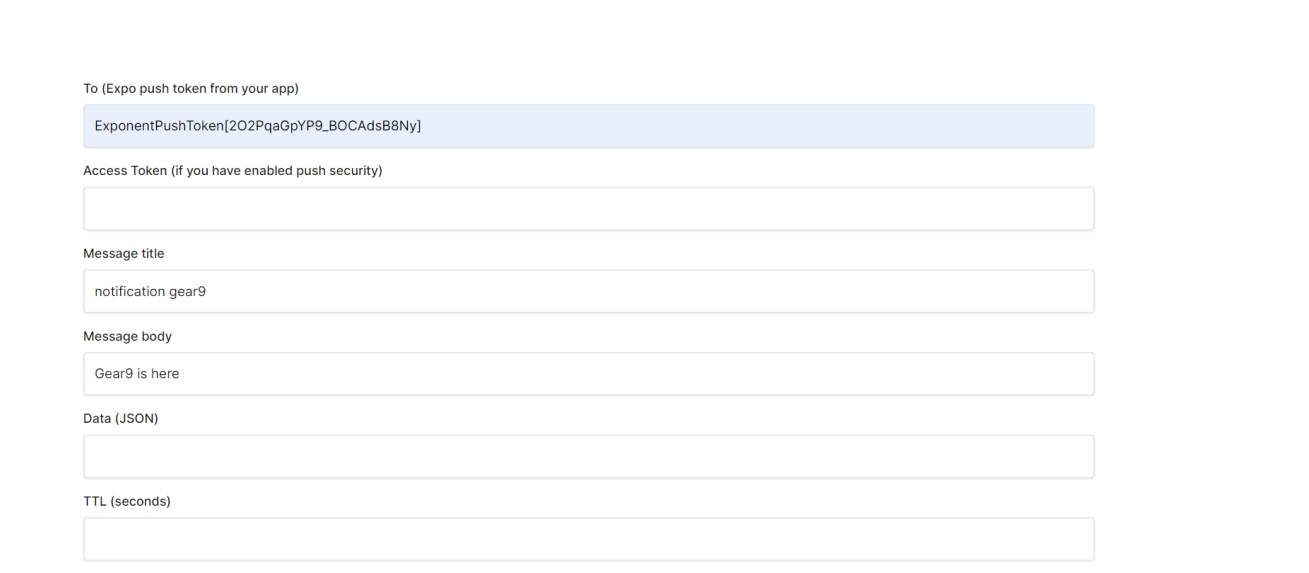


Figure 49: expo push notification tool

Screen :

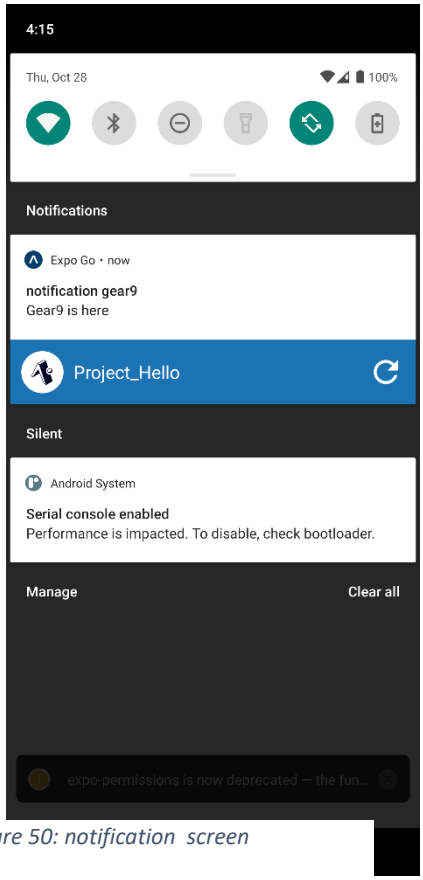


Figure 50: notification screen

Chapitre 21 : Polices Personnalisés

Exemples :

On Peut télécharger n'importe quelle type de police (par exemple à partir de google fonts)

```
import React from 'react';
import { View, Text } from 'react-native';
import AppLoading from 'expo-app-loading';
import { useFonts, Inter_900Black, } from '@expo-google-fonts/inter';
function AddPolice(props) {
    let [fontsLoaded] = useFonts({
        'Inter-black': require('./assets/fonts/IrishGrover-Regular.ttf'),
        'Festive-black': require('./assets/fonts/Festive-Regular.ttf'),

      });

      if (!fontsLoaded) {
        return <AppLoading />;
      } else {
        return (
          <View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>
            <Text style={{ fontFamily: 'Inter-black', fontSize: 40 }}>Irish Grover </Text>
            <Text style={{ fontFamily: 'Festive-black', fontSize: 40 }}>Festive </Text>
          </View>
        );
      }
}

export default AddPolice;
```

Figure 51: fonts

Screen :

Figure 52: fonts exemples

Chapitre 22 : RefreshControl

```
const [refreshing,setrefreshing] =useState(false)
const wait =(timeout)=>{
    return new Promise(resolve=>setTimeout(resolve,timeout))
}
const OnRefresh =()=>{
    setrefreshing(true)
    wait(2400).then(()=>setrefreshing(false))
}
return (
    <View style={styles.container}>
        <ScrollView
        refreshControl={
            <RefreshControl
            refreshing={refreshing}
            onRefresh={OnRefresh}
            />
        }
        >
        <Text style={styles.Title_Text}>Users' Jobs</Text>
        <Text style={{alignSelf:"center"}}> pull down to refresh</Text>
        {
            users.map((user,index)=>{
                return (
                    <View style={styles.user_view}>
                    <Text > name : {user.name}</Text>
                    <Text > job : {user.name}</Text>
                    </View>
                );
            })
        }
        </ScrollView>
    </View>
```

Figure 53: refresh control

Screens :

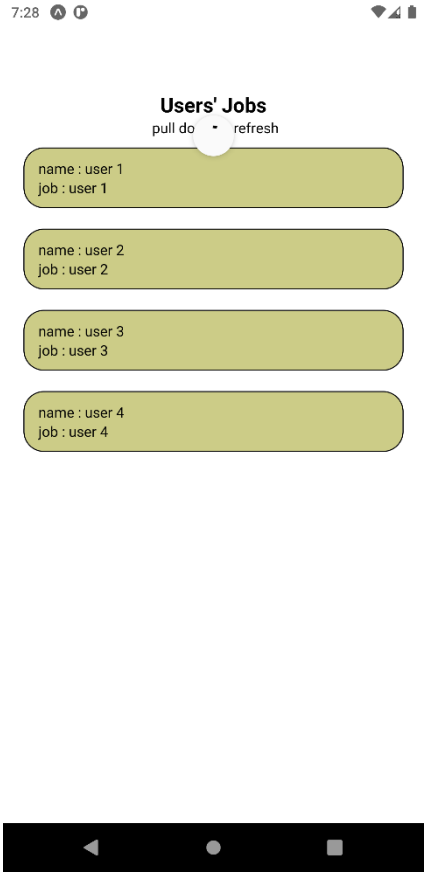


Figure 54:Refresh control screen

Chapitre 23 : Requêtes http

Syntaxe :

- Fetch(url , options) [.alors(…)[.catch(…)]]

Exemple : Posting application avec json server

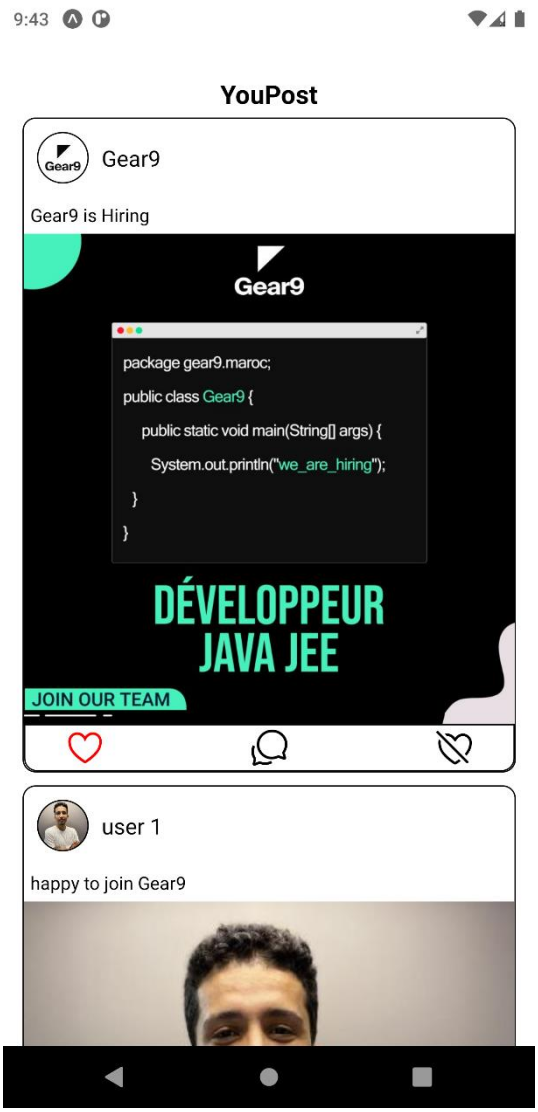
```
import { TouchableOpacity } from 'react-native-gesture-handler';
import { Ionicons } from '@expo/vector-icons';
function Requet(props) {
    const [data, setData] = useState([]);
    const [likes_num,setLikes]=useState(0);
    const Like =()=>{
        setLikes(likes_num+1)
    }
    const Dislike =()=>{
        if (likes_num===0)
        {
        setLikes(0)
        } else {
            setLikes(likes_num-1)
        }

    }
    const getPosts = async () => {
        try {
            const response =await fetch('http://192.168.0.106:3000/posts');
            const json = await response.json();
            setData(json);
        }catch(error) {
            console.log(error)
        }
    }
    useEffect(() => {
        getPosts();

      }, []);
    return (
        <ScrollView>
        <View style={styles.container}>
           <Text style={styles.Title_Text}>YouPost</Text>
           {
               data.map((post,index)=>{
                   return (
```

Screen :

Figure 55: fetch1

```
        <ScrollView>
        <View style={styles.container}>
           <Text style={styles.Title_Text}>YouPost</Text>
           {
               data.map((post,index)=>{
                   return (
                       <View style={styles.post_container}>
                           <View  style={styles.avatar_row}>
                               <Image  source={{uri : post.user_avatar}} style={styles.avatar}/>
                           <Text style={styles.user_text}>{post.user}</Text>

                           </View>
                           <Text style={styles.description_text}>{post.description}</Text>
                        <View>
                      <Image source={{uri : post.picture}}  style={styles.image_post}/>
                      </View>
                      <View style={styles.like_dislike_row} >
                         <TouchableOpacity onPress={()=>setLikes(likes_num+1)
}>
                           <Ionicons name="heart-outline" size={32} color="red" />

                           </TouchableOpacity>
                           <TouchableOpacity >
                           <Ionicons name="chatbubbles-outline" size={32} color="black" />

                           </TouchableOpacity>
                           <TouchableOpacity onPress={()=>Dislike()}>
                           <Ionicons name="heart-dislike-outline" size={32} color="black" />

                           </TouchableOpacity>
                      </View>
                       </View>

                   );
```

Figure 56: fetch 2


9:43
YouPost
Gear9
Gear9 is Hiring
Gear9
package gear9.maroc;
public class Gear9 {
public static void main(String[] args) {
System.out.println("we_are_hiring");
}
}
DÉVELOPPEUR
JAVA JEE
JOIN OUR TEAM
user 1
happy to join Gear9

Chapitre 24 : Test d'unité

Introduction :

Le tes unitaire est une pratique de test de bas niveau ou les plus petites unités ou composants du code son testés.

Exemples :

```
import React, { useEffect, useState } from 'react';
import { Button, StyleSheet, Platform,Text, View, Image, ScrollView } from 'react-native';

function Buttons(props) {
    return (
      <View style={styles.container}>
        <Text style={styles.Title_Text}>TESTING JEST {props.text}</Text>
        <Button title ="button for testing" />
      </View>
    );
}

export default Buttons;
const styles = StyleSheet.create({
    container:{marginTop:90,margin:10,marginBottom:10,},
    Title_Text: {fontSize:20,fontWeight:'bold',alignSelf:'center',margin : 20,},


})
```

Figure 57: Test for buttons component

```
import { exportAllDeclaration } from "@babel/types";
import React from "react";
import renderer from 'react-test-renderer';
import Buttons from "../../Components_testing/buttons";
describe ( '<Buttons/>', () =>{
    it('has 1 child ' ,()=>{
        const tree=renderer.create(<Buttons text="test from jset"/>).toJSON();
        expect(tree.children.length).toBe(2);

    });
}
);
```

Figure 58: buttonns.test.js

```
C:\Users\ahafdi\Desktop\react native\gear9_projects\Project_Hello>yarn test
yarn run v1.22.11
$ jest
 PASS  __tests__/components/buttons.test.js
  <Buttons/>
    √ has 1 child  (487 ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        2.192 s, estimated 3 s
Ran all test suites.
Done in 3.23s.
```

Figure 59: Test passed

Chapitre 25 : Mobx

Introduction :

State

Figure 60:Mobx + React Native

Management ou la gestion de états fait partie de développement des applications avec JavaScript. En particulier , les applications de React et React Native . dans cette documentation on va apprendre à utiliser la bibliothéque Mobx pour la gestion des états , comprendre les concepts de base .

C'est quoi Mobx :

Mobx est une bibliothèque éprouvée qui simplifier la gestion des états .

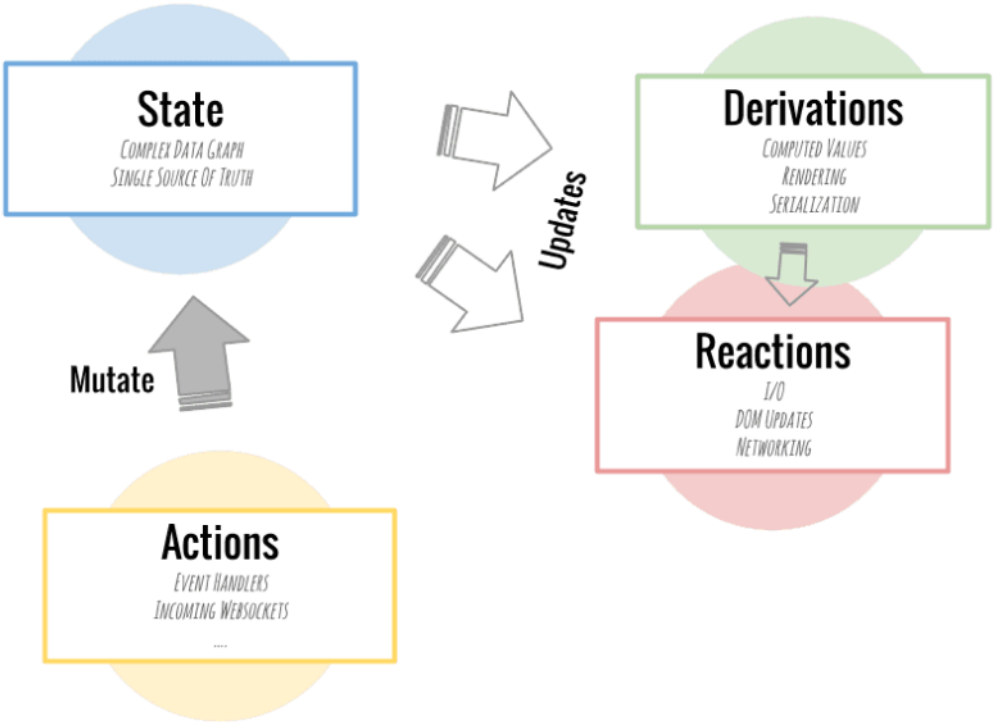


Figure 61: Mobx state architecture

Les Principes de Mobx :

- **STATE**

  STATE correspond aux données que contient l'application , il s'agit aussi le contenu de sa mémoire . ceci s'applique également à les composants .

- **DERIVATIONS**

  Dans Mobx ; tout ce qui peut être dérivé de l'état sans interactions est une dérivation .

- **ACTIONS**

  Contrairement , aux dérivations les actions sont tous les components et les fonctions qui mettent un changement sur les données.

Exemple 1 : une application simple des taches avec state management useState ()

```
function Task_SetState(props) {
    const [item,setItem]=useState('task 0');
    const [items,setItems]=useState(['task 1']);
    const RemoveTask =(index)=>{
        let CopyTasks=[...items];
        CopyTasks.splice(1,index);
        setItems(CopyTasks)
    }
    return (
        <ScrollView>
            <View style={styles.container}>
            <Text style={styles.Title_Text}> ToDo Tasks setstate</Text>
            <View style={styles.input_container}>
            <TextInput
            placeholder='write task'
            onChangeText={(text)=>setItem(text)}
            />
            <Button  title="add task"  onPress={()=>setItems([...items,item])}/>
            </View>
            {
                items.map((tasks,index)=>{
                    return (
                        <View style={styles.task_container}  >
                        <Text key={index}> {tasks} </Text>
                        <TouchableOpacity onPress={()=>RemoveTask(index)}>
                            <Ionicons name="trash-outline" size={32} color="red" />
                            </TouchableOpacity>

                        </View>
                        );
                })
            }

        </View>
        </ScrollView>
```

Figure 62: Task with UseState

Screen :

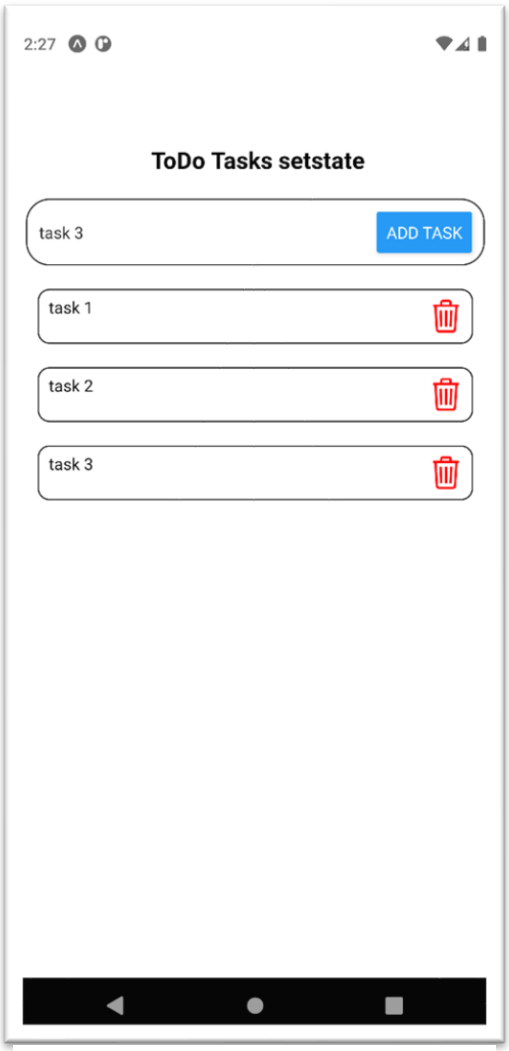


Figure 63: Tasks UseState screen

Exemple 2 :une application simple des taches avec state management Mobx

```
export default Task_Mobx  =  observer( ()=> {

    return (
        <ScrollView>
            <View style={styles.container}>
            <Text style={styles.Title_Text}> ToDo Tasks Mobx</Text>
            <View style={styles.input_container}>
            <TextInput
            placeholder='write task'
            onChangeText={(text)=>{new_store.tache=text}}
            />
            <Button  title="add task"  onPress={()=>new_store.addTaches(new_store.tache)}/>
            </View>
            {
                new_store.taches.map((tasks,index)=>{
                    return (
                        <View style={styles.task_container}  >
                        <Text key={index}> {tasks} </Text>
                        <TouchableOpacity onPress={()=>new_store.removetache(index)}>
                            <Ionicons name="trash-outline" size={32} color="red" />
                            </TouchableOpacity>

                        </View>
                        );
                    })
                }

            </View>
            </ScrollView>
```

Figure 64: Tasks With Mobx

Screen :

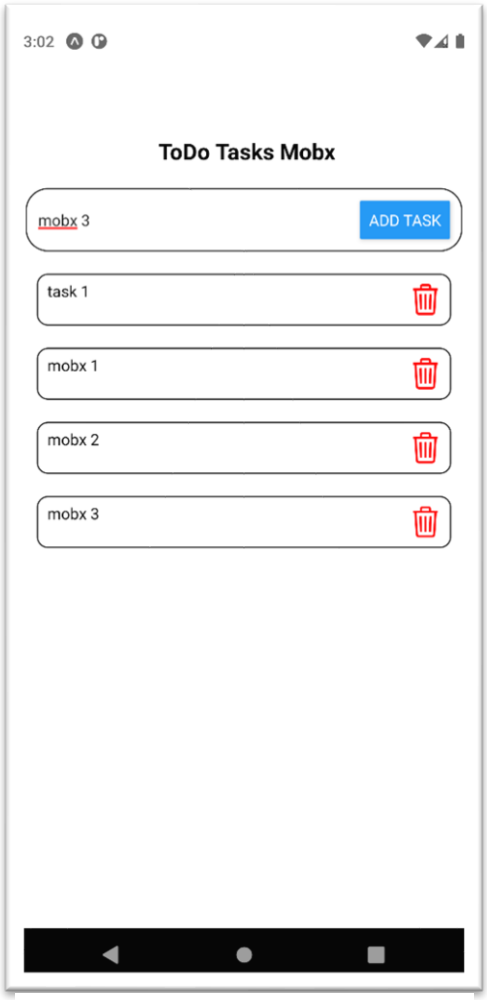


Figure 65: Mobx Tasks Screen

Exemple 3 : Exemple 2 :une application simple des taches avec state management MST (Mobx State Tree)

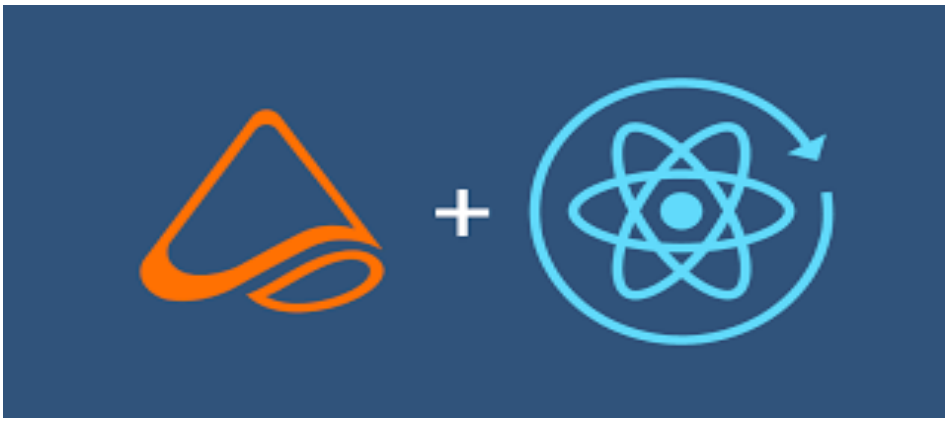


Figure 66: MST RN

C'est quoi MST :

Mobx State Tree est un bibliothèque de react et react native il permet de gérer les états mais mieux que Mobx car il permet de gérer les types de données et créer des modeles .

```
import {destroy, types} from "mobx-state-tree";
const Book = types.model ('Book',{
    title : types.string,
    author  : types.string,
})
const BookStore = types.model('Books',{
    books: types.array(Book)
})
.actions(self =>({
    addBook(book) {
        self.books.push(book)
    },
    removeBook(book) {
        destroy(book)
    }
}))
.create({
    books : [{title :"book 1",author : "author 1"},{title :"book 2",author : "author 2"}]
})
export default BookStore
```

Figure 67: Book Store

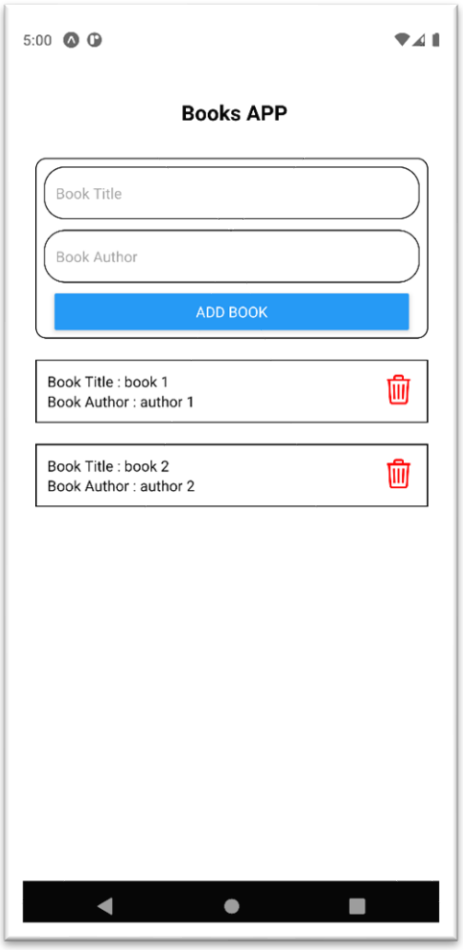
```
@Observer
export default class BookApp extends Component {
    state=initalestate;
    Value_Key (key,value){
        this.setState({
            [key]:value
        })
    };
    AjouterBook(){
        BookStore.addBook(this.state);
        this.setState(initalestate)
    };
    deleteBook(book) {
        BookStore.removeBook(book)
    }
    render() {
    return (
        <ScrollView>
            <View style={styles.container}>
            <Text style={styles.Title_Text}> Books APP</Text>
            <View style={styles.Book_Container}>
            <View style={styles.input_container}>
                <TextInput  placeholder="Book Title"
                onChangeText={(titre)=>this.Value_Key('title',titre)}
                />
            </View>
            <View style={styles.input_container}>
                <TextInput  placeholder="Book Author"
                onChangeText={(ecriv)=>this.Value_Key('author',ecriv)}

                />
            </View>
            <View style={styles.button_add}>
            <Button title ="Add Book"
            onPress={()=>this.AjouterBook()}
            />
            </View>
```

Figure 68: BookView 1

```
    return (
        <ScrollView>
            <View style={styles.container}>
            <Text style={styles.Title_Text}> Books APP</Text>
            <View style={styles.Book_Container}>
            <View style={styles.input_container}>
                <TextInput  placeholder="Book Title"
                onChangeText={(titre)=>this.Value_Key('title',titre)}
                />
            </View>
            <View style={styles.input_container}>
                <TextInput  placeholder="Book Author"
                onChangeText={(ecriv)=>this.Value_Key('author',ecriv)}

                />
            </View>
            <View style={styles.button_add}>
            <Button title ="Add Book"
            onPress={()=>this.AjouterBook()}
            />
            </View>
            </View>
            {
                books.map((livre,index)=>{
                    return (
                        <View style={styles.Book_Info}>
                        <View style={styles.book_view}>
                        <Text>Book Title : {livre.title}</Text>

                        <Text>Book Author : {livre.author}</Text>
                        </View>
                        <TouchableOpacity  style={styles.trash} onPress={()=>this.deleteBook(livre)}>
                            <Ionicons name="trash-outline" size={32} color="red" />
                            </TouchableOpacity>
                        </View>
                    );
```

Figure 69: BookView 2

Screen :


5:00
Books APP
Book Title
Book Author
ADD BOOK
Book Title : book 1
Book Author : author 1
Book Title : book 2
Book Author : author 2

Chapitre 26: Mini Projet 1 (Firebase users app)

Firebase users app : est une application simple d'ajouter /créer , supprimer et mettre à jour des comptes des utilisateurs c'est application pour admin pour gérer des comptes tout ça en utilisant la base de données firebase.

Code source :

Database/firebase.js :

```
// Import the functions you need from the SDKs you need
import firebase  from "firebase/app";
import "firebase/firestore";

// TODO: Add SDKs for Firebase products that you want to use
// https://firebase.google.com/docs/web/setup#available-libraries

// Your web app's Firebase configuration
// For Firebase JS SDK v7.20.0 and later, measurementId is optional
const firebaseConfig = {
  apiKey: "AIzaSyDz6remGPgHoO0gsZxx5QFjZzsTfIODoJI",
  authDomain: "crudbook-127e5.firebaseapp.com",
  projectId: "crudbook-127e5",
  storageBucket: "crudbook-127e5.appspot.com",
  messagingSenderId: "85964288116",
  appId: "1:85964288116:web:b6bd7b6a7b5544b8326121",
  measurementId: "G-XRPMR15E06"
};
// Initialize Firebase
firebase.initializeApp(firebaseConfig);
firebase.firestore();

export default firebase ;
```

Figure 70: firebase

Add User Screen :

```
class AddUserScreen extends Component {
  constructor (){
    super();
    this.dbRef=firebase.firestore().collection('users');/*connection to DB */
    this.state={
      name:'',
      email :'',
      mobile : '',
      isLoading : false
      /*this.state takes a column of data  from the db  */
    };
  }

  inputValueUpdate =(val,prop)=>{
    const state = this.state;
    state[prop]=val;
    this.setState(state);
    /*affecting the value inputed to this.state(column of data) */
  }

  storeUser () {
    if(this.state.name===''){
      alert("name is empty fill it !")
    } else {
      this.setState(
        {
          isLoading:true,

        }
      );
      this.dbRef.add({
        name : this.state.name,
        email : this.state.email,
```

Figure 71: adduserscreen1

```
      this.setState({
        name :'',
        email : '',
        mobile :'',
        isLoading : false,
      });
      this.props.navigation.navigate('UserScreen')
    }
    ).catch((err)=>{
      console.error("error found : ", err);
      this.setState({
        isLoading : false
      });

    }
    );
  }
}


render() {
  if (this.state.isLoading){
    return (
        <View style={styles.preloader}>
          <ActivityIndicator
          size="large" color ="#9E9E9E"
          />

        </View>
    )
  }
```

Figure 72: adduserscreen2

```
return (
  <ScrollView style={styles.container}>
    <View style={styles.input_container}>
      <View  style={styles.title_container}>
        <Ionicons name="people" size={23} color="blue" />
      <Text style={styles.title}> USERS APP</Text>
      </View>

     <View style={styles.inputgroup}>
      {/*name */}
      <Ionicons  name ="person" size={23}  style={{marginRight:10,}}  color="green"/>

      <TextInput
      multiline={true}
       placeholder={'Name'}
       value={this.state.name}
       onChangeText={(val)=>this.inputValueUpdate(val , 'name')}


      />

    </View>
    <View style={styles.inputgroup}>
      {/*email */}
      <Ionicons  name ="mail" size={23}  style={{marginRight:10,}} color="green"/>

      <TextInput
       placeholder={'Email'}
       value={this.state.email}
       onChangeText={(val)=>this.inputValueUpdate(val , 'email')}
      />
    </View>
    <View style={styles.inputgroup}>
    <Ionicons  name ="call" size={23}  style={{marginRight:10,}}  color="green" />
      {/*phone */}
```

Figure 73: adduserscreen3

```
      <Ionicons  name ="mail" size={23}  style={{marginRight:10,}} color="green"/>

      <TextInput
       placeholder={'Email'}
       value={this.state.email}
       onChangeText={(val)=>this.inputValueUpdate(val , 'email')}
      />
     </View>
     <View style={styles.inputgroup}>
     <Ionicons  name ="call" size={23}  style={{marginRight:10,}}  color="green" />
      {/*phone */}

      <TextInput
      placeholder={"Mobile"}
      value={this.state.mobile}
      onChangeText={(val)=>this.inputValueUpdate(val , 'mobile')}
      />
     </View>
      </View>


     <View  style={{flexDirection:'row',justifyContent:'space-evenly'}}>

      <Button
       title="Add User"
       color ='green'
       onPress={()=>this.storeUser()}
      />
      <Button
       title="List Users"
       color="blue"
       onPress={()=>this.props.navigation.navigate("UserScreen")}
      />
     </View>
    </ScrollView>
```

Figure 74: adduserscreen4

List Users :

```
class UserScreen extends Component {
  constructor (){
    super();
    this.firestoreRef=firebase.firestore().collection('users');
    this.state ={
      isLoading : true,
      userArr : []
    }
  }
  componentDidMount (){
    this.unsubscribe=this.firestoreRef.onSnapshot(this.getCollection);
  }
  componentWillUnmount(){
    this.unsubscribe();
  }
  getCollection =(querySnapshot)=>{
    const userArr =[];
    querySnapshot.forEach((res)=>
    {
      const{name,email,mobile}=res.data();
      userArr.push({
        key:res.id,
        res,
        name,
        email,
        mobile,
      });

    }
    );
    this.setState({
      userArr,
      isLoading : false,
```

Figure 75: list users 1

```
render() {
  if(this.state.isLoading){
    return(
      <View style={styles.preloader}>
        <ActivityIndicator size="large" color="#9E9E9E"/>
      </View>
    )
  }
  return (
    <ScrollView style={styles.container}>
       {
         this.state.userArr.map((item,i)=>
         {
           return (

            <ListItem key ={i} bottomDivider
            onPress={()=>{this.props.navigation.navigate('UserDetailScreen',{
            userKey : item.key}
            )}}
            >
              <ListItem.Content>
                <ListItem.Title>
                  {item.name}
                </ListItem.Title>
                <ListItem.Subtitle>
                  {item.email}
                </ListItem.Subtitle>
              </ListItem.Content>
            </ListItem>
          );
         }
         )}
```

Figure 76: list users 2

User Details Screen :

```
class UserDetailScreen extends Component {
  constructor (){
    super();
    this.state ={
      name:'',
      email :'',
      mobile :'',
      isloading : true,
    }
  }
  inputValueUpdate =(val,prop)=>{
    const state=this.state;
    state[prop]=val;
    this.setState(state);

  }
  updateUser (){
    this.setState(
      {
        isloading : true,

      }
    );
    const updateDBRef=firebase.firestore().collection('users').doc(this.state.key);
    updateDBRef.set({
      name : this.state.name,
      email : this.state.email,
      mobile : this.state.mobile,
    }).then((docRef)=>{
      this.setState({
        key :'',
        name : '',
        email : '',
        mobile : '',
        isloading : false,

      });
```

Figure 77: user screen details 1

```
  }).catch((error)=>{
    console.error("Error : ",error);
    this.setState({
      isloading : false,
    });
  });
}
deleteUser (){
  const dbRef = firebase.firestore().collection('users').doc(this.props.route.params.userKey);
  dbRef.delete().this((res)=>
  {
    console.log('Item removed from database')
    this.props.navigation.navigate('UserScreen');
  }
  )
}
openTwoButtonAlert=()=>{
  Alert.alert(
    'Delete User',
    'Are you sure?',
    [
      {text: 'Yes', onPress: () => this.deleteUser()},
      {text: 'No', onPress: () => console.log('No item was removed'), style: 'cancel'},
    ],
    {
      cancelable: true
    }
  );
}
```

Figure 78: user details screen 2

```
          <TextInput
          placeholder={"Email"}
          value={this.state.email}
          onChangeText={(val)=>this.inputValueUpdate(val, 'email')}
          />
        </View>
        <View style={styles.inputs}>
          <TextInput
          placeholder={"Mobile"}
          value={this.state.mobile}
          onChangeText={(val)=>this.inputValueUpdate(val, 'mobile')}
          />
        </View>
        </View>
        <View style={styles.buttons}>
          <Button
          title ="Update User"
          color="green"
          onPress ={()=>this.updateUser()}

          />
           <Button
          title="Delete User"
          color="red"
          onPress ={()=>this.openTwoButtonAlert()}
          />


        </View>


    </ScrollView>
```

Figure 79: user screen details 3

Screens :

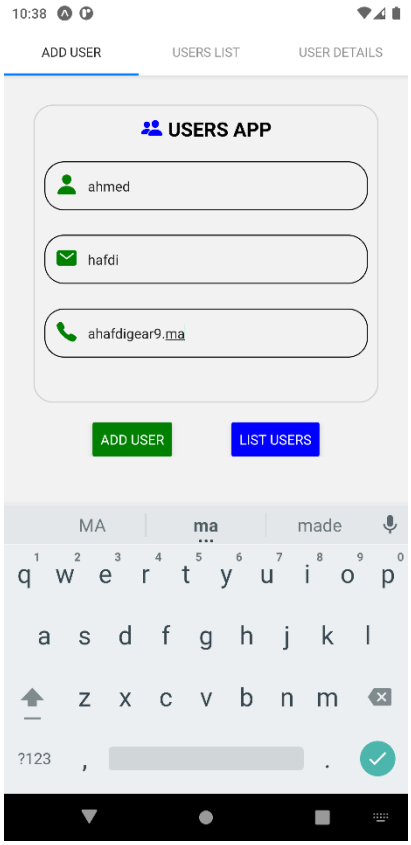


Figure 81: add user

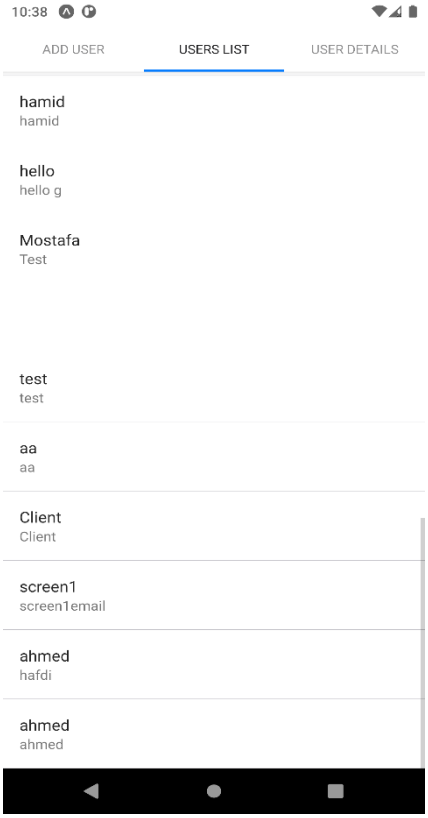


Figure 82: list users

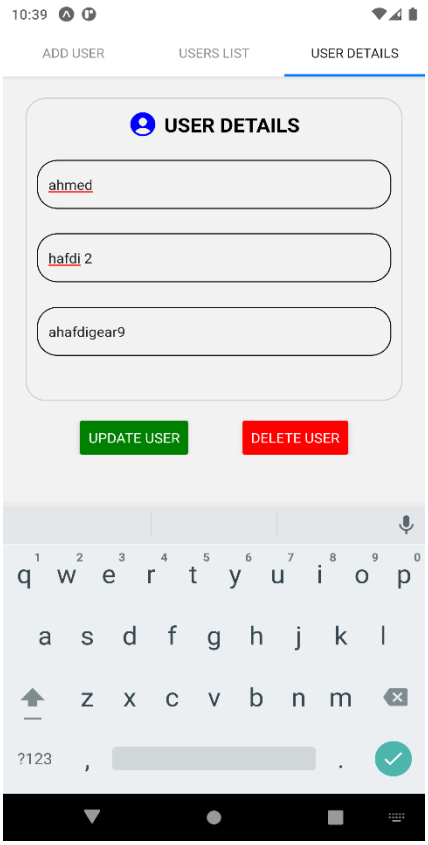


Figure 80: user details

Chapitre 27: Mini Projet 2 (Tasks app)

Voir Chapitre 25

Chapitre 28: Ignite Boilerplate un générateur de react native

Introduction :

React-Native est un outil génial , mais démarrer un nouveau projet avec react native n'est pas du tout facile .

Il demande beaucoup des outils à installer et des bibliothèques …etc , en plus de ça il demande de faire des règles pour les structure des dossiers .

Alors l'installation de tous ça va prendre beaucoup de temps .

Figure 83: Infinite Red Logo

c'est pour cela **Infinite Red** (software company since 2010) a crée **Ignite** un générateur pour React Native avec Boilerplates , plugins ..etc , l'equipe de Ignite a lancé plusieurs version de **Ignite**

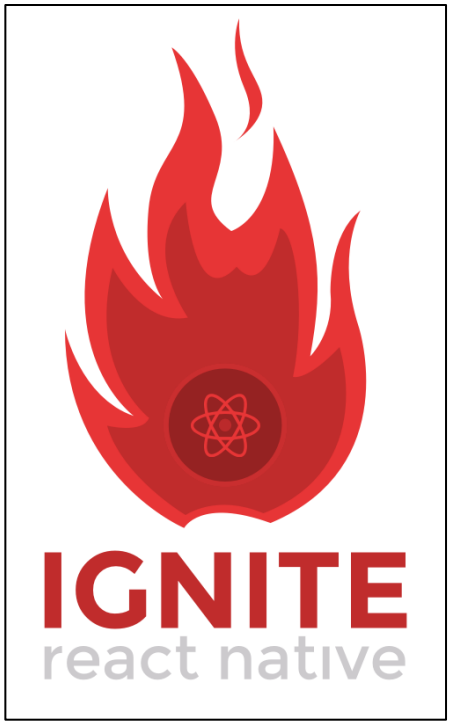


Figure 84 ignite logo

Ignite est bien connu dans le monde React Native. C'est une collection de toutes les opinions d'**Infinite Red** sur la pile, les modèles et les packages en un seul endroit. Cela permet généralement aux équipes d'économiser environ deux à quatre semaines sur le front-end de leurs projets React Native.

Les applications d'Ignite incluent :

- ***React Native***
- ***React Navigation 5***
- ***MobX-State-Tree (Why not Redux?)***
- ***MobX-React-Lite***
- ***TypeScript***
- ***AsyncStorage (integrated with MST for restoring state)***
- ***apisauce (to talk to REST servers)***
- ***Flipper-ready***
- ***Reactotron-ready (and pre-integrated with MST)***
- ***Supports Expo (and Expo web) out of the box***
- ***And more!***

Installation de Ignite (ignite setup sur windows) :

- Installation de Ignite Commande Line (ignite cli) :

  yarn add ignite-cli

Création d'un projet Ignite :

```
# for vanilla React Native
npx ignite-cli new PizzaApp
# or for Expo-powered:
npx ignite-cli new PizzaApp --expo
# to provide a custom bundle identifier (Android only):
npx ignite-cli new PizzaApp --bundle=com.infinitered.pizzaapp
```

Chapitre 28: Premier Projet React-Native par Ignite bowser

Pour créer un projet React-Native avec Ignite on lance la commande suivante :

npx ignite new ProjectName bowser

N.B : la création prend beaucoup de temps (presque 30 min)

Screen :

```
C:\Users\ahafdi\Desktop\react native>npx ignite new FirstPorject bowser
Info: You provided more than one argument for <projectName>. The first one (FirstPorject) will be used and the rest are
ignored.


  . . . . . .  Ignite  . . . . . .

  Creating FirstPorject using Ignite 7.6.0
  Powered by Infinite Red - https://infinite.red
  Using ignite-cli
  Bundle identifier: com.FirstPorject


  Igniting app
  3D-printing a new React Native app
  \ Unboxing npm dependencies
```

Figure 85: ignite first project

Apres on lance le projet sur un émulateur de Android par la commande suivante :

react-native run-android

Screen :

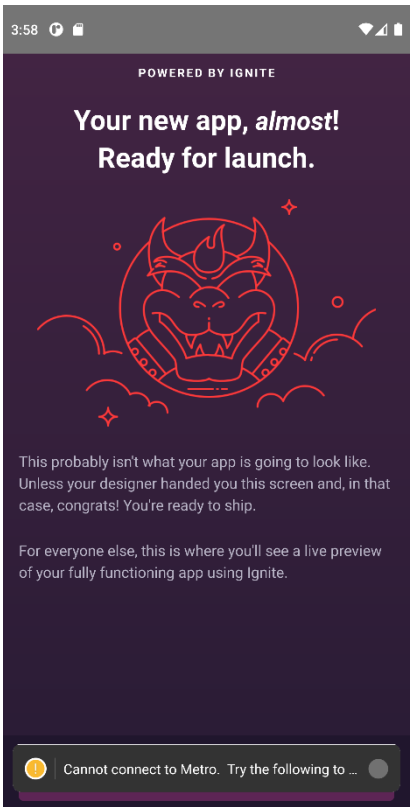


Figure 86: welcome screen ignite

La structure du projet Ignite :

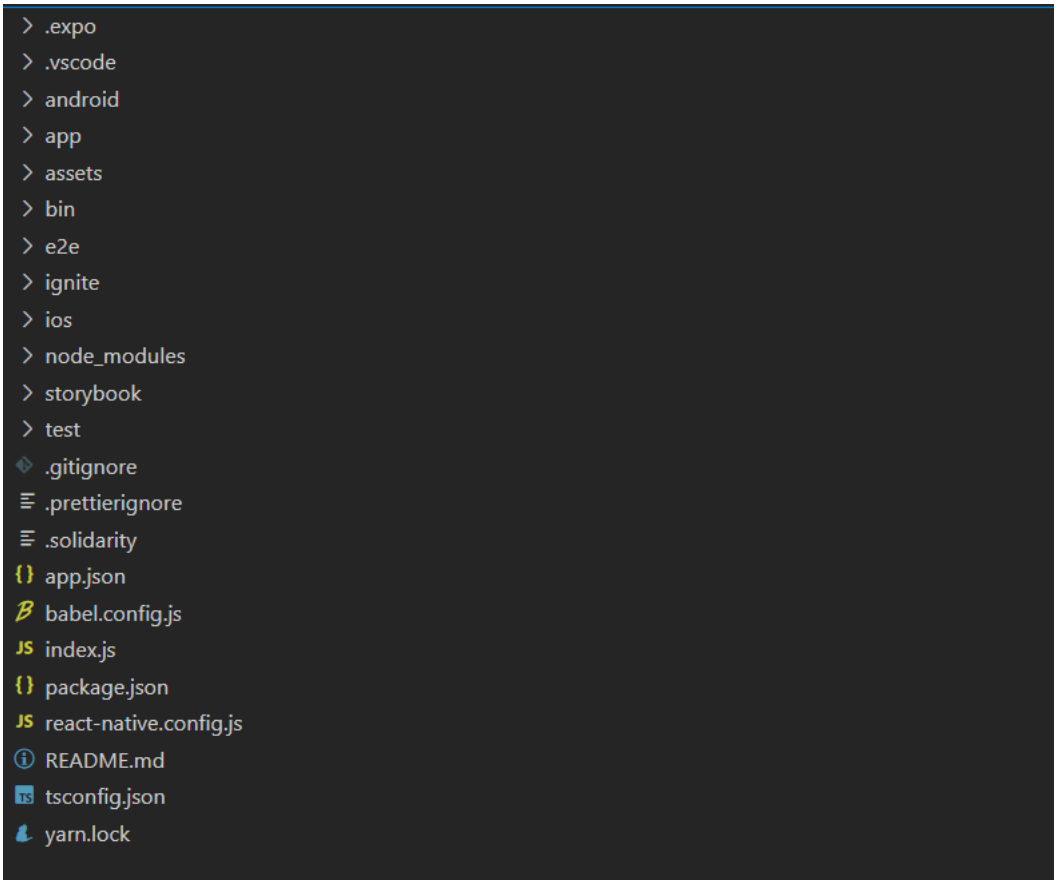


Figure 87: project -structure

Le dossier qui est comme le cœur ou le noyau du projet est le dossier « app »

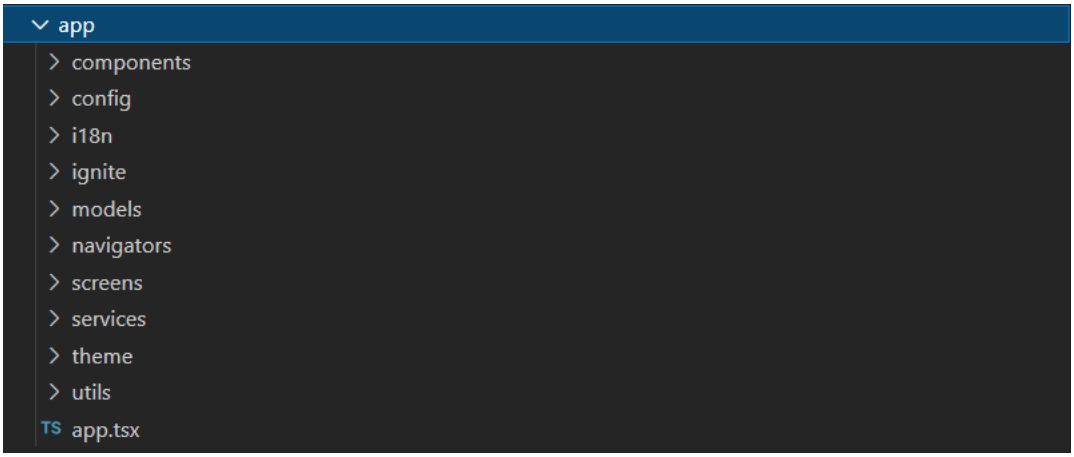


Figure 88: app structure

La génération des screens avec ignite : on lance la commande « **npx ignite-cli screen profile** » :

SCREENS :

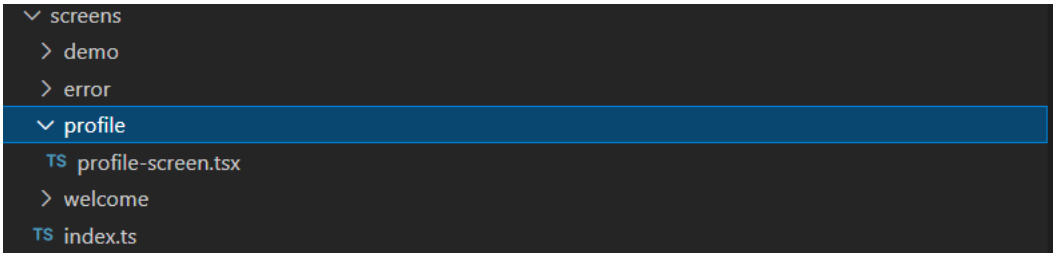


Figure 89: screens ignite

La génération des models avec ignite : on lance la commande « **npx ignite-cli model profile** » :

MODELS :

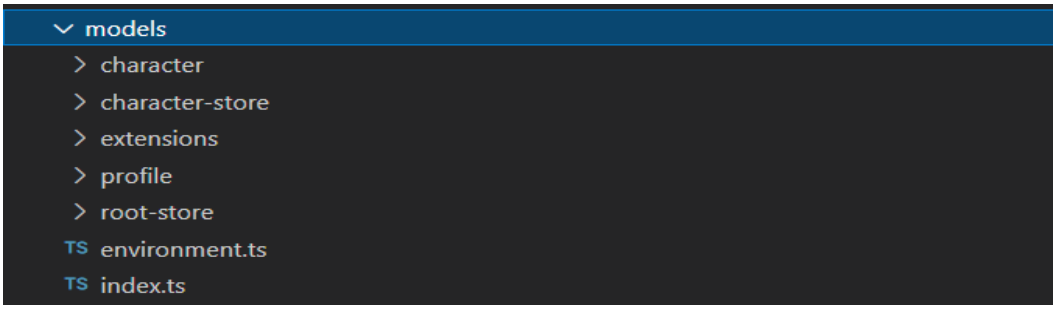


Figure 90: models

La génération des components avec ignite : on lance la commande « **npx ignite-cli component profile** » :

COMPONENTS :

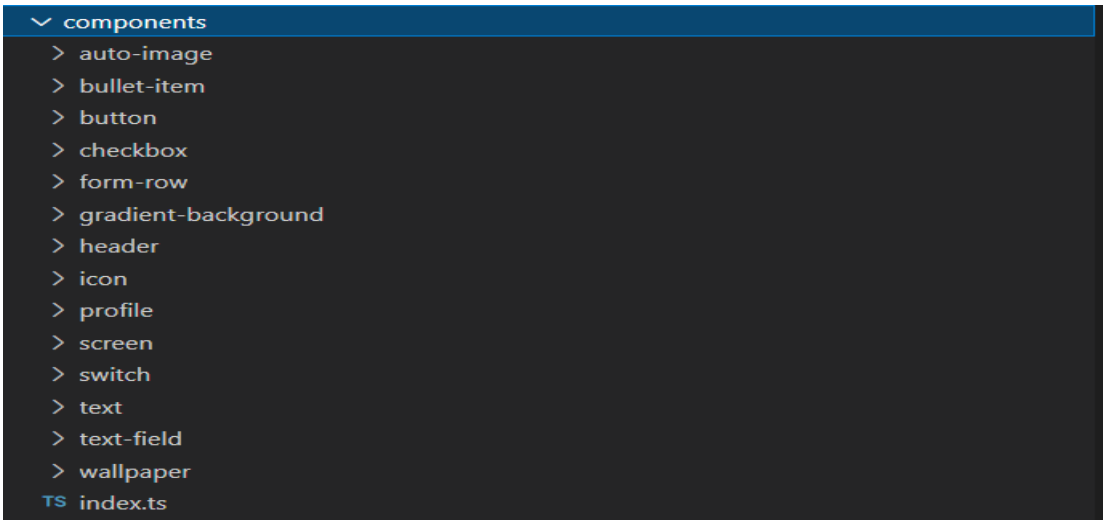


Figure 91:components

Navigation en Ignite :

Le générateur ignite offre un dossier speciale pour la navigation entre les screens dans React-Native

C'est le dossier « Navigators »

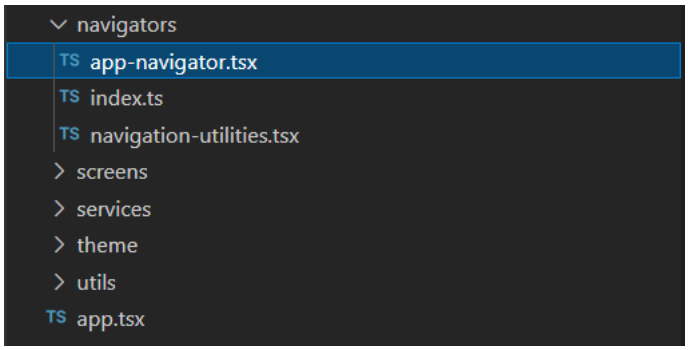


Figure 92: navigators

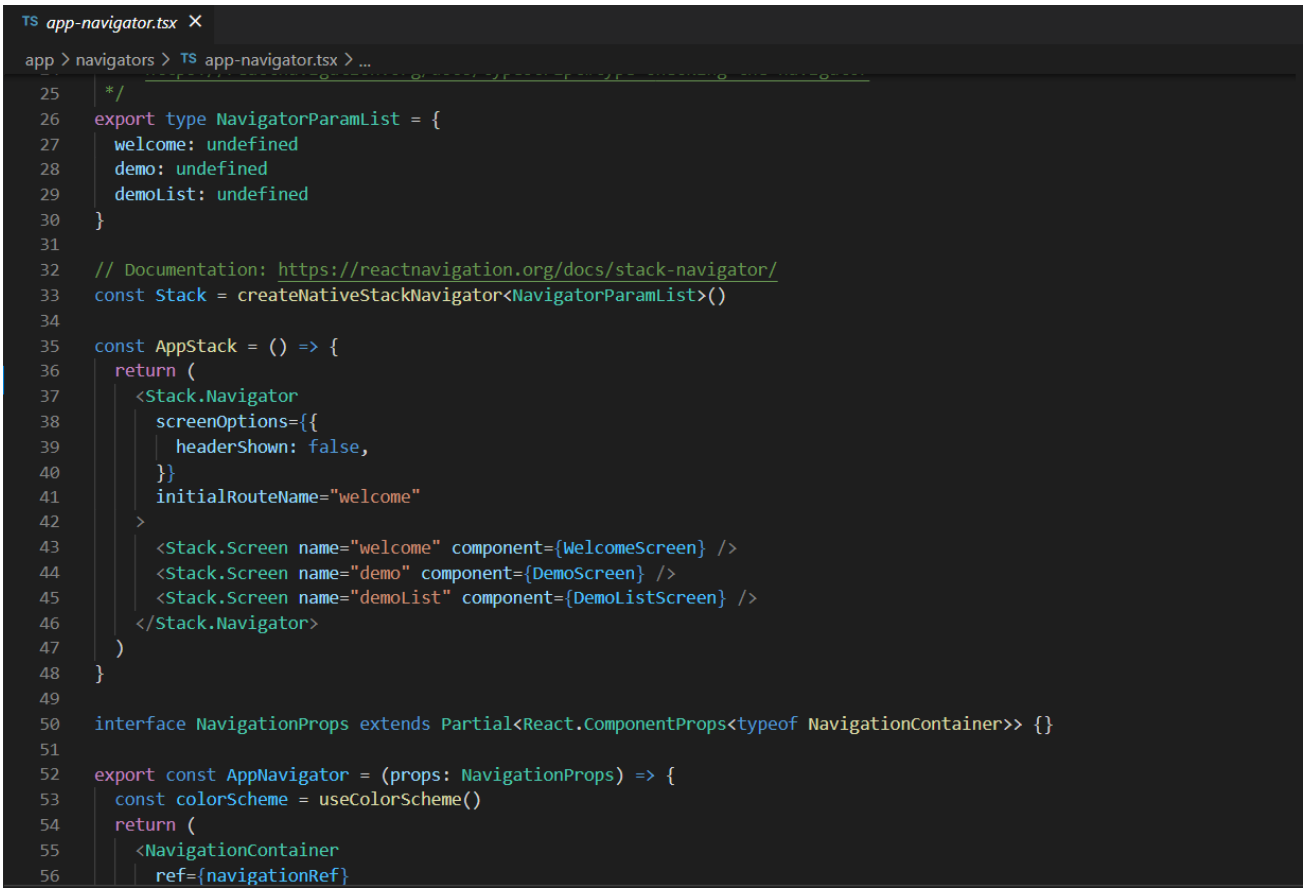


Figure 93:app navigator

Alors dans le fichier **app-navigator.ts** Ignite offre un stack screen pour la navigation par défaut .

Les services de Ignite :

Ignite offre un dossier spécial pour les services de API et http requêtes :

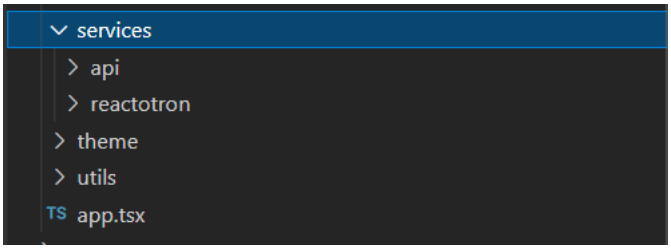


Figure 94: services

Comme le dossier API :

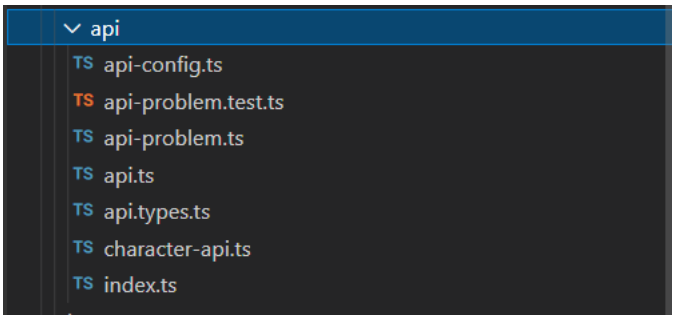


Figure 95: api

Dans le dossier api : il ya plusieurs fichiers mai les plus importants sont : le fichier **api-config** c'est pour les fonctions de http requêtes (comme les fonctions pour fetcher , poster , créer et supprimer les données via les requêtes) et le fichier **api-types** c'est pour définir les types des modelés et les interfaces et finalement les fichier **api** c'est pour définir le 'url de l'api .

Comme le dossier Reactotron :

L'inspection des applications est désormais aussi simple qu'elle a toujours été censée l'être. Inspectez sans effort les applications mobiles React JS et React Native avec **Reactotron**, une application de bureau gratuite d'Infinite Red.

Reactotron setup :

Local :

On installe l'application [Reactotron]

In-app :

Pour les projets de ignite la configuration existe par défaut dans le dossier Reactotron

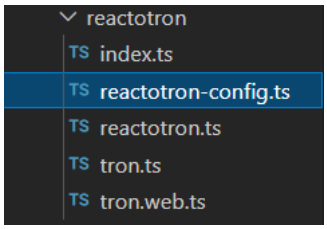


Figure 96: reactotron dossier

Lancer Reactotron :

On lance L'application desktop de Reactotron , après on lance la commande :

```
adb reverse tcp:9090 tcp:9090
```

c'est pour connecter avec l'application et le projet .

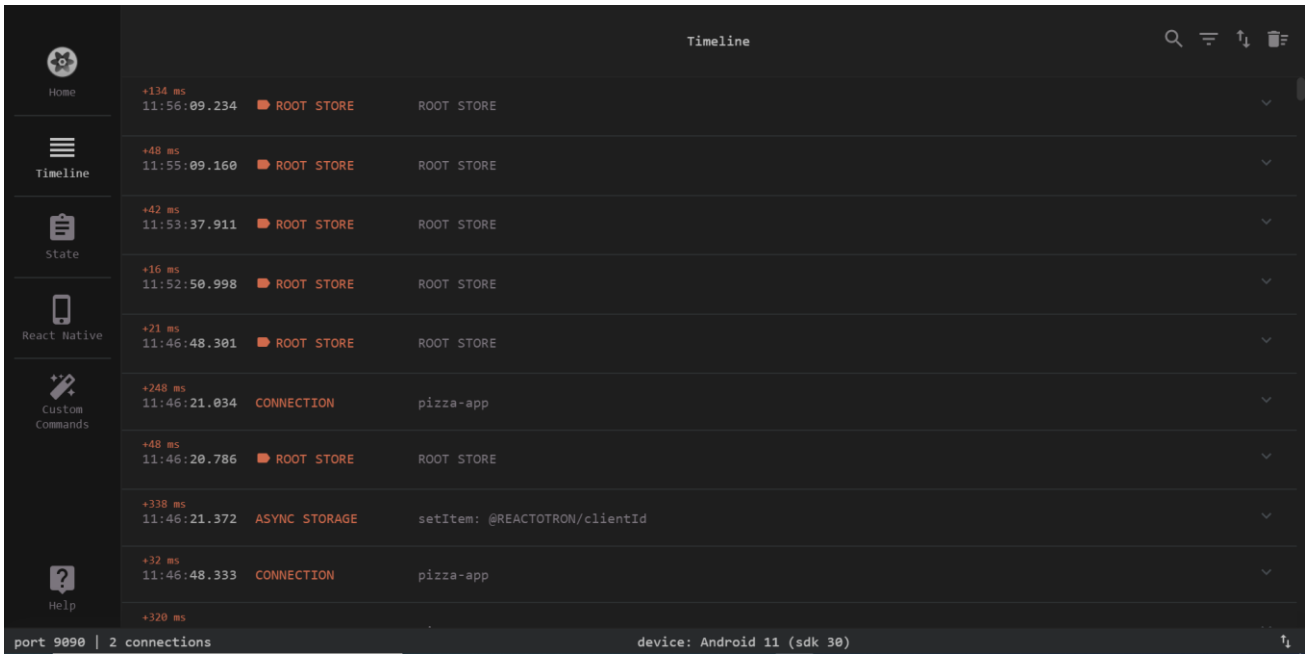


Figure 97: reactotron connection

Chapitre 29: Mini Projet Avec Ignite et Service Api

Explication : c'est application React-Native pour se connecter et s'inscrire et visualiser les produits .

1) Backend : On lance le server qui géré l'api de cette application :

```
const fs = require('fs')
const bodyParser = require('body-parser')
const jsonServer = require('json-server')
const jwt = require('jsonwebtoken')

const server = jsonServer.create()
const router = jsonServer.router('./database.json')
const userdb = JSON.parse(fs.readFileSync('./users.json', 'UTF-8'))

server.use(bodyParser.urlencoded({extended: true}))
server.use(bodyParser.json())
server.use(jsonServer.defaults());

const SECRET_KEY = '123456789'

const expiresIn = '1h'

// Create a token from a payload
function createToken(payload){
  return jwt.sign(payload, SECRET_KEY, {expiresIn})
}

// Verify the token
function verifyToken(token){
  return  jwt.verify(token, SECRET_KEY, (err, decode) => decode !== undefined ?  decode : err)
}

// Check if the user exists in database
function isAuthenticated({email, password}){
  return userdb.users.findIndex(user => user.email === email && user.password === password) !== -1
}

// Register New User
server.post('/auth/register', (req, res) => {
  console.log("register endpoint called; request body:");
  console.log(req.body);
  const {email, password} = req.body;
```

Figure 98: server 1

```
  if(isAuthenticated({email, password}) === true) {
    const status = 401;
    const message = 'Email and Password already exist';
    res.status(status).json({status, message});
    return
  }

fs.readFile("./users.json", (err, data) => {
    if (err) {
      const status = 401
      const message = err
      res.status(status).json({status, message})
      return
    };

    // Get current users data
    var data = JSON.parse(data.toString());

    // Get the id of last user
    var last_item_id = data.users[data.users.length-1].id;

    //Add new user
    data.users.push({id: last_item_id + 1, email: email, password: password}); //add some data
    var writeData = fs.writeFile("./users.json", JSON.stringify(data), (err, result) => {  // WRITE
        if (err) {
          const status = 401
          const message = err
          res.status(status).json({status, message})
          return
        }
    });
});

// Create token for new user
  const access_token = createToken({email, password})
  console.log("Access Token:" + access_token);
  res.status(200).json({access_token})
```

Figure 99: server 2

```
server.post('/auth/login', (req, res) => {
  console.log("login endpoint called; request body:");
  console.log(req.body);
  const {email, password} = req.body;
  if (isAuthenticated({email, password}) === false) {
    const status = 401
    const message = 'Incorrect email or password'
    res.status(status).json({status, message})
    return
  }
  const access_token = createToken({email, password})
  console.log("Access Token:" + access_token);
  res.status(200).json({access_token})
})

server.use(/^(?!\/auth).*$/,  (req, res, next) => {
  if (req.headers.authorization === undefined || req.headers.authorization.split(' ')[0] !== 'Bearer') {
    const status = 401
    const message = 'Error in authorization format'
    res.status(status).json({status, message})
    return
  }
  try {
    let verifyTokenResult;
     verifyTokenResult = verifyToken(req.headers.authorization.split(' ')[1]);

     if (verifyTokenResult instanceof Error) {
       const status = 401
       const message = 'Access token not provided'
       res.status(status).json({status, message})
       return
     }
     next()
  } catch (err) {
    const status = 401
    const message = 'Error access_token is revoked'
```

Figure 100: server 3

```
    let verifyTokenResult;
     verifyTokenResult = verifyToken(req.headers.authorization.split(' ')[1]);

     if (verifyTokenResult instanceof Error) {
       const status = 401
       const message = 'Access token not provided'
       res.status(status).json({status, message})
       return
     }
     next()
  } catch (err) {
    const status = 401
    const message = 'Error access_token is revoked'
    res.status(status).json({status, message})
  }
})

server.use(router)

server.listen(8000,'192.168.0.106', () => {
  console.log('Run Auth API Server')
})
```

Figure 101: server 4

Pour lancer le server on utilise la commande :

npm run start-auth

2) On crée les interfaces : UI pour s'inscrire

```
import React, { useState } from "react"
import { observer } from "mobx-react-lite"
import { ImageStyle, SafeAreaView, TextStyle, ViewStyle } from "react-native"
import { Button, Profile, Screen, Text } from "../../../components"
import { useNavigation } from "@react-navigation/native"
import { color } from "../../../theme"
import metrics from "../../../theme/metrics"
import { useStores } from "../../../models"

export const SignupScreen = observer(function SignupScreen() {
  // Pull in one of our MST stores
  // const { someStore, anotherStore } = useStores()
  const {ProfileStore}=useStores();
  // Pull in navigation via hook
   const navigation = useNavigation()
  const [profileInputs,SetProfileInputs]=useState({
    profileEmail : "",
    profilePassword:""
  })
  return (
    <Screen style={ROOT} preset="scroll">
        <SafeAreaView>
      <Text preset="header" text="SINGUP To Gear9" style={HEADER_STYLE} />
      <Profile  textinput="Enter Your Email"
      onChangeText={(text)=>profileInputs.profileEmail=text}
      />
     <Profile  textinput="Enter Your Password"
      onChangeText={(text)=>profileInputs.profilePassword=text}
      secureTextEntry={true}
     />
     <Button text="SIGN UP" style={BUTTON_SIGNIN} textStyle={TextButton}
     onPress={()=>{SetProfileInputs({
       profileEmail:profileInputs.profileEmail,
       profilePassword:profileInputs.profilePassword,
     });
     ProfileStore.Register(profileInputs.profileEmail,profileInputs.profilePassword);
     if(ProfileStore.status===200){
```

Figure 102: Register 1

```
       navigation.navigate("signin")
       ProfileStore.setStatus(123)
     }

     }
   }
      />
       </SafeAreaView>
     </Screen>
   )
})

const ROOT: ViewStyle = {
  backgroundColor: color.palette.black,
  flex: 1,
}
const HEADER_STYLE : ViewStyle = {
  alignSelf : 'center',
  marginVertical:metrics.heightPercentageToDP(5),
}

const BUTTON_SIGNIN :ViewStyle={
  alignSelf:'center',
  margin: metrics.widthPercentageToDP(2),
  backgroundColor:'black',
  borderRadius:10,
}
const TextButton:TextStyle={
  fontSize:20,
}
const LOGO : ImageStyle={
  alignSelf:'center',
  margin:metrics.widthPercentageToDP(5),
  borderRadius:20,
  width:metrics.widthPercentageToDP(90),
  height:metrics.heightPercentageToDP(25),
}
```

Figure 103: Register 2

Screen :

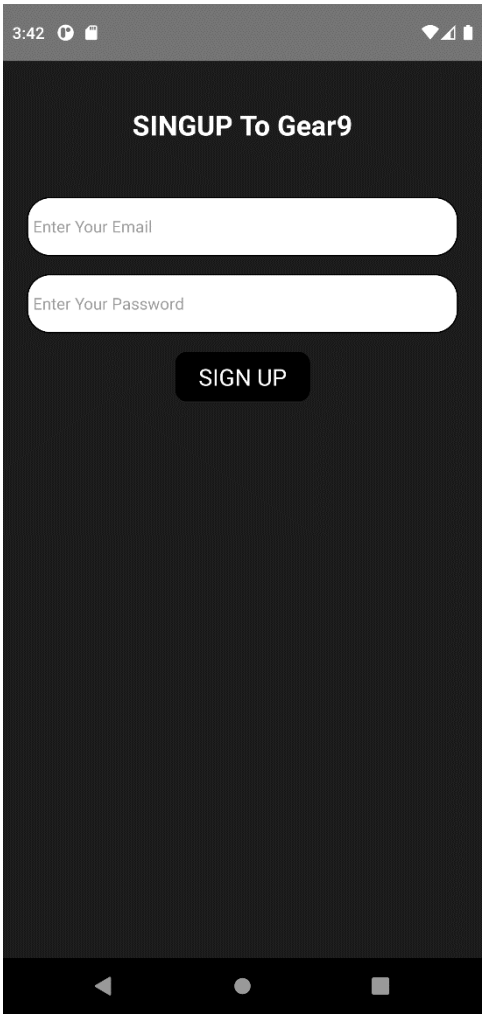


Figure 104: SIGNUP Gear9

3) On crée les interfaces : UI pour se connecter

```
import React, { useEffect, useState } from "react"
import { observer } from "mobx-react-lite"
import { Image, ImageStyle, SafeAreaView, TextStyle, TouchableOpacity, ViewStyle } from "react-native"
import { Button, Profile, Screen, Text } from "../../../components"
// import { useNavigation } from "@react-navigation/native"
// import { useStores } from "../../models"
import { color } from "../../../theme"
import metrics from "../../../theme/metrics"
import { useNavigation } from "@react-navigation/core"
import { useStores } from "../../../models"


const gear9_logo=require("../../../../assets/images/gear9logo.jpg");

export const SigninScreen = observer(function SigninScreen() {
  // Pull in one of our MST stores
  // const { someStore, anotherStore } = useStores()
  const {ProfileStore}=useStores();

  // Pull in navigation via hook
   const navigation = useNavigation()
  const [profileInputs,SetProfileInputs]=useState({
    profileEmail : "",
    profilePassword:""
  })
  useEffect(()=>{
  })
  return (
    <Screen style={ROOT} preset="scroll">
        <SafeAreaView>
      <Text preset="header" text="SINGIN To Gear9" style={HEADER_STYLE} />
      <Image source={gear9_logo} style={LOGO}/>
     <Profile  textinput="Enter Your Email"
      onChangeText={(text)=>profileInputs.profileEmail=text}
      />
     <Profile  textinput="Enter Your Password"
      onChangeText={(text)=>profileInputs.profilePassword=text}
```

Figure 105: Login 1

```
      />
      <Button text="SIGN IN" style={BUTTON_SIGNIN} textStyle={TextButton}
      onPress={()=>{SetProfileInputs({
        profileEmail:profileInputs.profileEmail,
        profilePassword:profileInputs.profilePassword,
      });
      ProfileStore.Login(profileInputs.profileEmail,profileInputs.profilePassword);
     if(ProfileStore.status===200){
       navigation.navigate("profile")

     }
     ProfileStore.setStatus(123)


     }}
      />
      <TouchableOpacity onPress={()=>navigation.navigate("signup")}>
      <Text>Dont you have an account , Register</Text>

      </TouchableOpacity>
       </SafeAreaView>
     </Screen>
   )
 })
 const ROOT: ViewStyle = {
   backgroundColor: color.palette.black,
   flex: 1,
 }
 const HEADER_STYLE : ViewStyle = {
   alignSelf : 'center',
   marginVertical:metrics.heightPercentageToDP(5),
 }
```

Figure 106: login 2

```
const ROOT: ViewStyle = {
  backgroundColor: color.palette.black,
  flex: 1,
}
const HEADER_STYLE : ViewStyle = {
  alignSelf : 'center',
  marginVertical:metrics.heightPercentageToDP(5),
}

const BUTTON_SIGNIN :ViewStyle={
  alignSelf:'center',
  margin: metrics.widthPercentageToDP(2),
  backgroundColor:'black',
  borderRadius:10,
}
const TextButton:TextStyle={
  fontSize:20,
}
const LOGO : ImageStyle={
  alignSelf:'center',
  margin:metrics.widthPercentageToDP(5),
  borderRadius:20,
  width:metrics.widthPercentageToDP(90),
  height:metrics.heightPercentageToDP(25),
}
```

Figure 107: login 3

Screen :

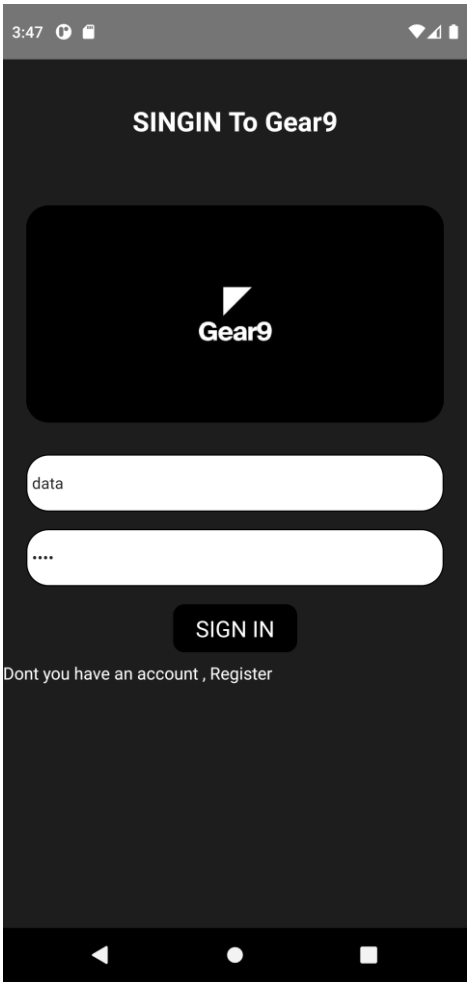


Figure 108: login to gear9

4) On crée les interfaces : UI pour profile screen

```
import React, { useEffect, useState } from "react"
import { observer } from "mobx-react-lite"
import { Image, ImageStyle, TextInput, TextStyle, View, ViewStyle } from "react-native"
import { Button, Profile, Screen, Text } from "../../components"
import { useNavigation } from "@react-navigation/native"
// import { useStores } from "../../models"
import { color } from "../../theme"
import metrics from "../../theme/metrics"
import { useStores } from "../../models"
import { SafeAreaView } from "react-native-safe-area-context"
import { TouchableOpacity } from "react-native-gesture-handler"

const gear9_logo=require("../../../assets/images/gear9logo.jpg");

export const ProfileScreen = observer(function ProfileScreen() {
  // Pull in one of our MST stores
  // const { someStore, anotherStore } = useStores()
  const {ProfileStore}=useStores();
  // Pull in navigation via hook
   const navigation = useNavigation()

  useEffect(()=>{
    //console.log(ProfileStore.Login("nilson@email.com","nilson"))
   //ProfileStore.GetPorducts("eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6ImhlbGxvIiwicGFzc3dvcmQiOiJoZWxsbyIsImlhdCI6MTYzNjk4NzA5MywiZXhwIjoxNjM2OTkwNjkzfQ.SEta5u6
   //console.log("hello")
      //console.log(ProfileStore.getEmail)
  })
  return (
    <Screen style={ROOT} preset="scroll">
      <SafeAreaView>
      <Text preset="header" text="Welcome To Gear9" style={HEADER_STYLE} />
      <View style={PROFILE_ITEM}>
        <Text>Your Email : {ProfileStore.email} </Text>
        </View>
        <View style={PROFILE_ITEM}>
        <Text>Your Password : {ProfileStore.password}  </Text>
        </View>
```

Figure 109: profile 1

```
    <Screen style={ROOT} preset="scroll">
      <SafeAreaView>
      <Text preset="header" text="Welcome To Gear9" style={HEADER_STYLE} />
      <View style={PROFILE_ITEM}>
        <Text>Your Email : {ProfileStore.email} </Text>
        </View>
        <View style={PROFILE_ITEM}>
        <Text>Your Password : {ProfileStore.password}  </Text>
        </View>
        <View style={PROFILE_ITEM}>
        <Text>Your Token : {ProfileStore.token}  </Text>
        </View>

        <Button style={BUTTON_SIGNIN} textStyle={TextButton}  text="LOGOUT" onPress={()=>{navigation.navigate("signin");ProfileStore.setStatus(123)}} />
      </SafeAreaView>

    </Screen>
  )
})
const ROOT: ViewStyle = {
  backgroundColor: color.palette.black,
  flex: 1,
}
const HEADER_STYLE : ViewStyle = {
  alignSelf : 'center',
  marginVertical:metrics.heightPercentageToDP(5),
}

const PROFILE_ITEM:ViewStyle={
  borderWidth:1,
  borderColor:"white",
  borderRadius:5,
  marginHorizontal:metrics.widthPercentageToDP(3),
  marginVertical:metrics.heightPercentageToDP(1),
}
```

Figure 110: profile 2

```
const ROOT: ViewStyle = {
  backgroundColor: color.palette.black,
  flex: 1,
}
const HEADER_STYLE : ViewStyle = {
  alignSelf : 'center',
  marginVertical:metrics.heightPercentageToDP(5),
}

const PROFILE_ITEM:ViewStyle={
  borderWidth:1,
  borderColor:"white",
  borderRadius:5,
  marginHorizontal:metrics.widthPercentageToDP(3),
  marginVertical:metrics.heightPercentageToDP(1),
}
const BUTTON_SIGNIN :ViewStyle={
  alignSelf:'center',
  margin: metrics.widthPercentageToDP(2),
  backgroundColor:'black',
  borderRadius:10,
}
const TextButton:TextStyle={
  fontSize:20,
}
```

Figure 111: profile 3

Screen :

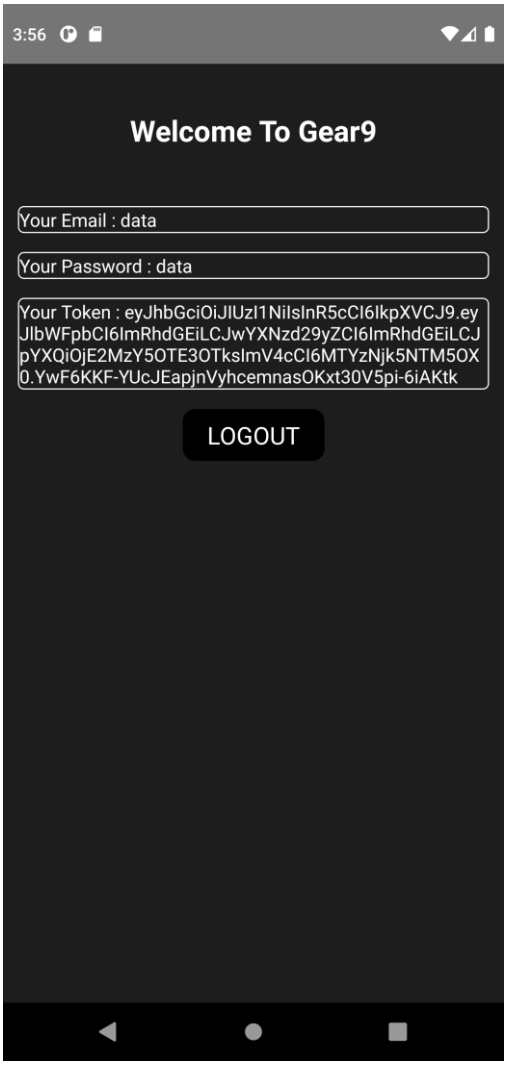


Figure 112: profile screen

5) Définir les Types :

```
import { GeneralApiProblem } from "./api-problem"
import { Character } from "../../models/character/character"

export interface User {
  id: number
  name: string
}

export type GetUsersResult = { kind: "ok"; users: User[] } | GeneralApiProblem
export type GetUserResult = { kind: "ok"; user: User } | GeneralApiProblem
export type ProfileType = {status : number,token : string,email : string, password:string} | GeneralApiProblem
export type ProductType = {status : number , products:any} |GeneralApiProblem
export type GetCharactersResult = { kind: "ok"; characters: Character[] } | GeneralApiProblem
export type GetCharacterResult = { kind: "ok"; character: Character } | GeneralApiProblem
```

Figure 113: Types

6) On crée les fonctions : fonction pour se connecter

```
async ProfileLogin(email : string , password : string): Promise<Types.ProfileType> {
  const data_input = {email,password}
  const response : ApiResponse<any>=await this.apisauce.post('/auth/login',data_input);
  if (!response.ok) {
    const problem = getGeneralApiProblem(response)
    if (problem) return problem
  }

  try {
      const ProfileResult=response.data
      return {status: response.status,token : ProfileResult.access_token,email: data_input.email, password :data_input.password}
  }catch {
    return {kind : "bad-data"}
  }
}
```

Figure 114: Profile Login

```
.actions((self)=>({
  Login : flow (function* (email : string, password: string){
    const api = new Api()
    api.setup()
    yield api.ProfileLogin(email, password).then((response : any)=>{
      self.setToken(response.token)
      self.setEmail(response.email)
      self.setPassword(response.password)
      self.setStatus(response.status)
    })

  }),
```

Figure 115: Login function

7) On crée les fonctions : fonction pour s'inscrire :

```
async ProfileRegister(email : string , password : string): Promise<Types.ProfileType> {
  const data_input = {email,password}
  const response : ApiResponse<any>=await this.apisauce.post('/auth/register',data_input);
  if (!response.ok) {
    const problem = getGeneralApiProblem(response)
    if (problem) return problem
  }

  try {
      const ProfileResult=response.data
      return {status: response.status,token : ProfileResult.access_token,email: data_input.email, password :data_input.password}
  }catch {
    return {kind : "bad-data"}
  }
}
```

Figure 116: profile register

```
Register : flow (function * (email : string , password : string ){
  const api = new Api()
  api.setup()
  yield api.ProfileRegister(email, password).then((response: any)=>{

    self.setToken(response.token)
    self.setEmail(response.email)
    self.setPassword(response.password)
    self.setStatus(response.status)

  })
}),
```

Figure 117: Register

8) On crée les fonctions : fonction pour les produits

```
async ProductFetch(token : string ): Promise<Types.ProductType> {

  this.apisauce.headers["Authorization"]='Bearer ' +token
  const response : ApiResponse<any>=await this.apisauce.get('/products');
  if (!response.ok) {
    const problem = getGeneralApiProblem(response)
    if (problem) return problem
  }
  try {
      const ProductList=response.data
      return {status: response.status,products:ProductList.map(
        prod=>ProductModel.create({
          id : prod.id,
        name : prod.name,
        cost : prod.cost,
        quantity: prod.quantity,
        locationId: prod.locationId,
        familyId : prod.familyId,
        })

      )}
  }catch {
    return {kind : "bad-data"}
  }
```

Figure 118: fetch products

```
  GetProducts : flow (function * (token : string){
    const api = new Api()
    api.setup()
    yield api.ProductFetch(token).then((response : any)=>{

      self.setStatus(response.status);
      self.setProducts_id(response.products)
    })
  })
})) // eslint-disable-line @typescript-eslint/no-unused-vars
```

Figure 119: Get Products

Screen :

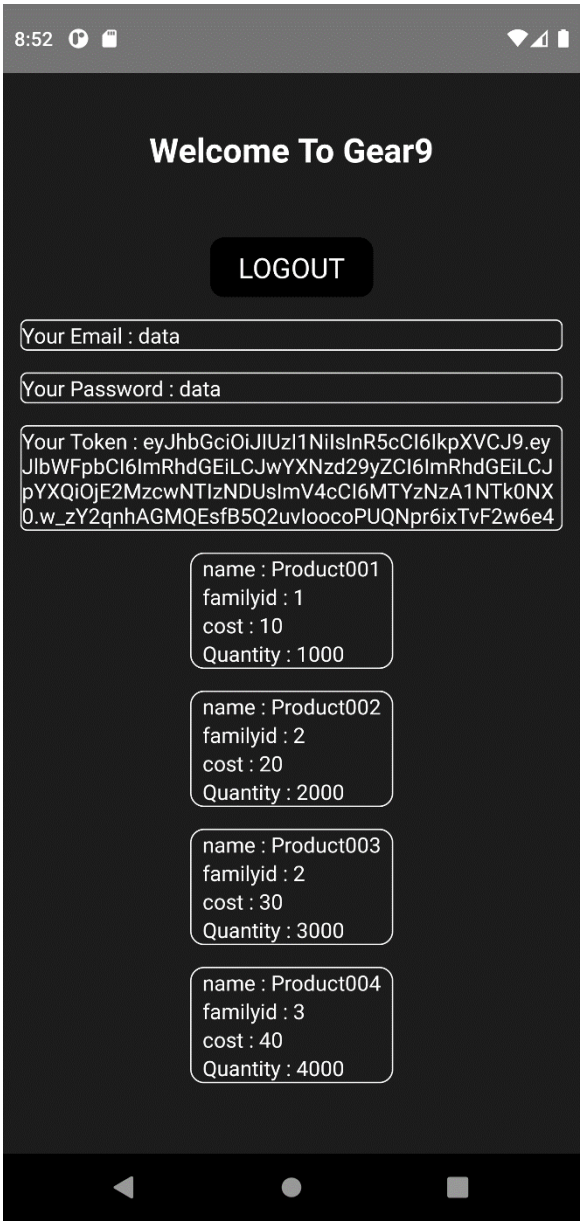


Figure 120: products screen

Chapitre 30: EXERCICE D'APPLICATION PROJET RED

Introduction :

Le projet RED est une application mobile pour la digitalisation d'une compétition pour les épiciers.

Les taches demandés :

C'est la réalisation de la partie front-end à partir d'XD (adobe xd) .

La structure du projet :

Le projet est composé de quatre grandes parties :

- **Accueil**
- **Mes Points**
- **Mes Gains**
- **Mes Challenges**

1) Authentification et SplasheScreen

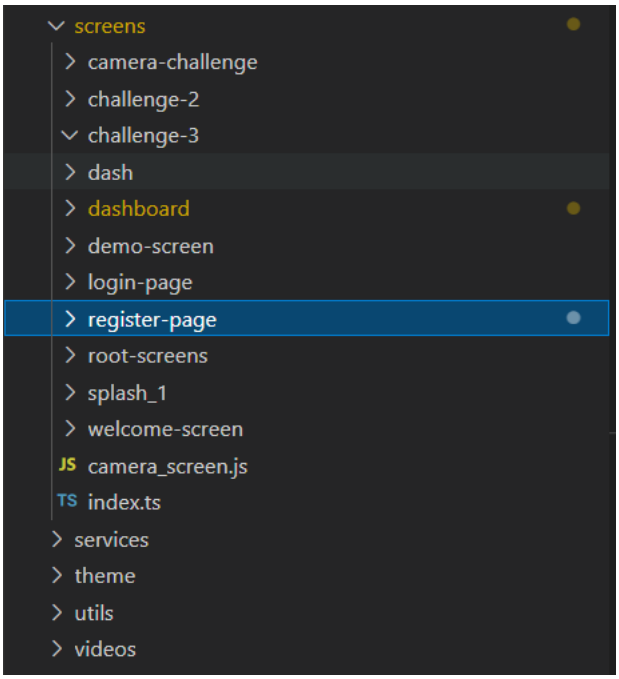


Figure 121: Architecture du projet

- **Splash Screen**

```
import React from "react"
import { observer } from "mobx-react-lite"
import { Image, ImageStyle, View, ViewStyle } from "react-native"
import { Screen, Text } from "../../components"
// import { useNavigation } from "@react-navigation/native"
// import { useStores } from "../../models"
import { color } from "../../theme"

const ROOT: ViewStyle = {
  backgroundColor: color.palette.primary,
  flex: 1,
}
const LOGO: ImageStyle={
  alignSelf: "center",
  maxWidth: "100%",


}
const WELCOME_TEXT : ViewStyle={

}

export const SplashScreen = observer(function SplashScreen() {
  // Pull in one of our MST stores
  // const { someStore, anotherStore } = useStores()

  // Pull in navigation via hook
  // const navigation = useNavigation()
  return (
    <Screen style={ROOT} preset="scroll">
      <Text preset="header" text="" />
      <Image source={require("./logo.png")}  style={LOGO} />
      <View style={WELCOME_TEXT}>
```

Figure 122: splash 1

```
export const SplashScreen = observer(function SplashScreen() {
  // Pull in one of our MST stores
  // const { someStore, anotherStore } = useStores()

  // Pull in navigation via hook
  // const navigation = useNavigation()
  return (
    <Screen style={ROOT} preset="scroll">
      <Text preset="header" text="" />
      <Image source={require("./logo.png")}  style={LOGO} />
      <View style={WELCOME_TEXT}>
        <Text style={{textAlign : 'center',fontSize :40}}>مرحبا بكم </Text>
        <Text style={{textAlign : 'center',fontSize :40}}>  في تطبيق  <Text style={{textAlign : 'center',fontSize :40,fontWeight:'bold'}}>زيد</Text> </Text>



      </View>


    </Screen>
  )
})
```

Figure 123: splash 2

Screen :

Figure 124: splash screen

- **Login Page**

```
import React,{useState,useEffect} from "react"
import { observer } from "mobx-react-lite"
import {  Image, ImageBackground, ImageStyle, TextInput, TextStyle, View, ViewStyle } from "react-native"
import { Button, Screen, Text } from "../../components"
import { useNavigation } from "@react-navigation/native"
// import { useStores } from "../../models"
import { color } from "../../theme"
import { Ionicons } from '@expo/vector-icons';
import {  TouchableOpacity } from "react-native-gesture-handler"
import ToggleSwitch from 'toggle-switch-react-native'
export const back_ground = require("../../images/loginBG.png")
export const logo_red = require("../../images/logo.png")
export const user_icon=require("../../images/user3.png")

const ROOT: ViewStyle = {
  flex:1,


}
const BACKGROUND_IMAGE : ImageStyle ={

  width: '100%', height: '100%',

}
const LOGO:ImageStyle = {
  alignSelf:'center',
  width: 200,
```

Figure 125: login1

```
export const LoginPageScreen = observer(function LoginPageScreen() {
  // Pull in one of our MST stores
  // const { someStore, anotherStore } = useStores()

  // Pull in navigation via hook
  const navigation = useNavigation()

  const [value,onChangeText]=useState()
  const [visible,setVisibility]=useState(false)
  const icon_pwd=visible? 'eye-off':'eye';


  return (
    <Screen style={ROOT} >

      <ImageBackground source={back_ground} style={BACKGROUND_IMAGE}>
      <Image source={logo_red} style={LOGO}/>
      <Text style={{textAlign:'center',fontSize:23,color:color.palette.black,fontWeight:'bold'}}>مرحبا بكم </Text>

      <Text style={{textAlign:'center',fontSize:23,color:color.palette.yellow,fontWeight:'bold',marginBottom:5}}> في تطبيق ريد</Text>


      <View style={LOGIN_VIEW}>
        <View style={input_login}>
        <Ionicons name="person" size={20} color ={color.palette.black}/>
                <TextInput  placeholder="   اسم المستعمل"/>
        </View>
        <View style={input_login}>
        <Ionicons name="lock-closed" size={20} color={color.palette.black}  />
                <TextInput  placeholder="كلمة المرور"
                onChangeText={text=>{onChangeText(text)}}
                value={value}
```

Figure 126: login2

```
        <Ionicons name={icon_pwd}  size={20} color={color.palette.black}  style={{marginRight:180}} onPress={()=>setVisibility(!visible)} />

        </View>
        <View  style={PASSWORD_FORGET}>
        <TouchableOpacity><Text style={{color:color.palette.orangeDarker,fontWeight:'bold',fontSize:13,}}>هل نسيت كلمة المرور؟</Text></TouchableOpacity>
        <ToggleSwitch
          isOn={false}
          onColor="#EB9918"
          offColor="#F6BC50"
          label="تذكر كلمة المرور "
          labelStyle={{ color:color.palette.orangeDarker, fontWeight: "bold",fontSize:13, }}
          size="small"
          onToggle={(isOn) =>{isOn=true}}
  />


        </View>
        <View style={BUTTON_LOGIN}>
          <Button text="دخول "
          style={{backgroundColor:color.palette.orangeDarker,borderRadius:20,width:"50%",}}
          textStyle={{textAlign:'center',fontSize:15,fontWeight:'bold'}}
          onPress={()=>navigation.navigate("menu")}
          />

        </View>



      </View>
      <Text style={{textAlign:'center',fontSize : 20,marginTop:30,fontWeight :'bold'}}>معندكش حساب؟</Text>
```

Figure 127: login3

```
    </View>
    <Text style={{textAlign:'center',fontSize : 20,marginTop:30,fontWeight :'bold'}}>معندكش حساب؟</Text>

    <View  style={Button_Container} >
    <Button
    style={BUTTON_REGISTER}
    text='تسجل معانا هنا'
    textStyle={BUTTON_TEXT}
    onPress={()=>navigation.navigate("register")}

    />
    <Text style={{textAlign :'center',color:color.palette.black,fontSize:14,fontWeight:'bold',marginTop:90,}}>2020 @ RED</Text>
    </View>
    <View style={FOOTER}>
      <Text style={FOOTER_TEXT}>الشروط و البنود</Text>
      <Text style={FOOTER_TEXT}>|</Text>
      <Text style={FOOTER_TEXT}>اشعار بسرية التعاقد</Text>
      <Text style={FOOTER_TEXT}>|</Text>
      <Text style={FOOTER_TEXT}>سياسة الكوكيز</Text>
      <Text style={FOOTER_TEXT}>|</Text>
      <Text style={FOOTER_TEXT}>معلومات الشركة</Text>

    </View>

</ImageBackground>

  </Screen>
)
})
```

Figure 128: login4

Screen :

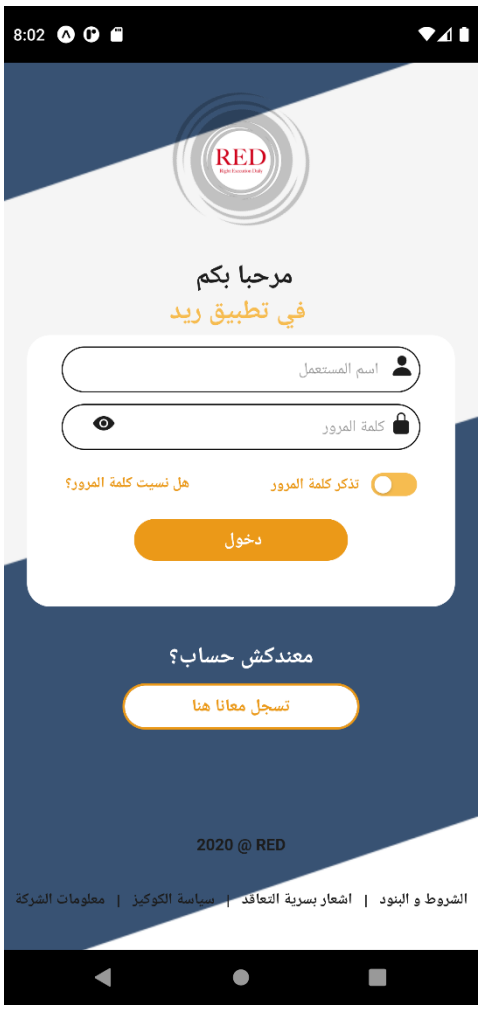


Figure 129: login page

- **Register Page**

```
import React, { useState } from "react"
import { observer } from "mobx-react-lite"
import { ImageBackground, ViewStyle,ImageStyle, Image, View, TextStyle } from "react-native"
import { Screen, Text } from "../../components"
import { useNavigation } from "@react-navigation/native"
// import { useStores } from "../../models"
import { color } from "../../theme"
import { ProgressSteps, ProgressStep } from 'react-native-progress-steps';
import { TextInput } from "react-native-gesture-handler"
import { Ionicons } from '@expo/vector-icons';
import SelectDropdown from 'react-native-select-dropdown'



export const back_ground = require("../../images/loginBG.png")
export const logo_red = require("../../images/logo.png")
export const bag_icon='../../icons/bag.svg'

const ROOT: ViewStyle = {
  backgroundColor: color.palette.black,
  flex: 1,
  //flexDirection:'column'
}
const BACKGROUND_IMAGE : ImageStyle ={

  width: '100%', height: '100%',

}
```

Figure 130: register 1

```
  return (
    <Screen style={ROOT}  >
      <ImageBackground source={back_ground} style={BACKGROUND_IMAGE}>
      <Image source={logo_red} style={LOGO}/>
      <Text style={{textAlign:'center',fontSize:23,color:color.palette.black,fontWeight:'bold'}}>تسجل معانا هنا</Text>
      <View style={REGISTER_VIEW}>
      <ProgressSteps {...progressStepsStyle}  >
        <ProgressStep   nextBtnStyle  ={buttonTextStyle} nextBtnText='التالي  ->' nextBtnTextStyle={{fontSize:20,color:color.palette.white,fontWeight:'bold'}}>

          <View style={input}>
        <Ionicons name="person-sharp" size={20} color={color.palette.black} style={{marginLeft:10}}  />
            <TextInput  placeholder="الاسم الكامل "/>

        </View>
        <View style={input}>
        <Ionicons name="call" size={20} color={color.palette.black} style={{marginLeft:10}}  />
            <TextInput  placeholder="الهاتف "/>

        </View>
        <View style={input}>
        <Ionicons name="mail" size={20} color={color.palette.black} style={{marginLeft:10}}  />
            <TextInput  placeholder="البريد الإلكتروني  "/>

        </View>
        <View style={input}>
        <Ionicons name="location" size={20} color={color.palette.black} style={{marginLeft:10}}  />
            <TextInput  placeholder="اسم المدينة "/>

        </View>
```

Figure 131: register 2

Screen :

Figure 133: register 1

Figure 132: register 2

- **La partie Accueil**

Architecture de Dashboard(menu) :

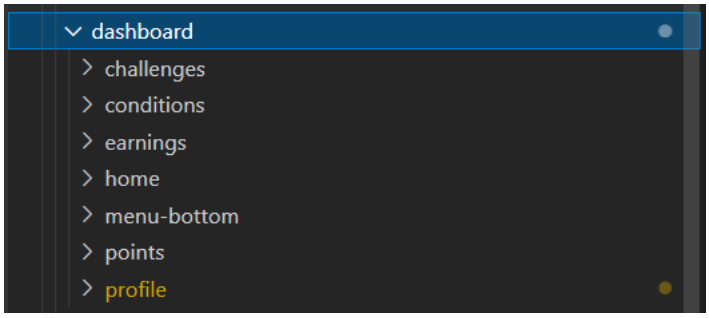


Figure 134: dashboard

- **La partie Accueil (home)**

```
import React, { useState } from "react"
import { observer } from "mobx-react-lite"
import {Dimensions, Image, ImageStyle, TextStyle, TouchableOpacity, View, ViewStyle } from "react-native"
import { Screen, Text } from "../../../components"
 import { useNavigation } from "@react-navigation/native"
// import { useStores } from "../../models"
import { Video } from 'expo-av';

import { color } from "../../../theme"
import { ScrollView } from "react-native-gesture-handler";
import Ionicons from "@expo/vector-icons/build/Ionicons"
export const home_icon = require("../../../images/home-active.png")

const ROOT: ViewStyle = {
  backgroundColor: color.palette.offWhite,
  flex: 1,
}
const BOX_SQUARES : ViewStyle = {
  flexDirection:'row',
  justifyContent:'center',


}
const BOX_SQUARE : ViewStyle = {
  height:140,
  width:140,
  backgroundColor:"white",
  marginHorizontal:10,
  marginTop:20,
  borderRadius:15,
  shadowColor: "#000",
shadowOffset: {
  width: 0,
  height: 2,
},
```

Figure 135 : home 1

```
export const points_box = require("../../../images/point.png")
export const earnings_box = require("../../../images/feather-gift.png")
export const conditions_box = require("../../../images/terms-and-conditions.png")
export const challenges_box = require("../../../images/challenge.png")
export const video_index=require("../../../videos/index.mp4")
export const HomeScreen = observer(function HomeScreen() {
  // Pull in one of our MST stores
  // const { someStore, anotherStore } = useStores()

  // Pull in navigation via hook
  const navigation = useNavigation()
  //const Tab = createBottomTabNavigator();



  return (
    <Screen style={ROOT} preset="scroll">
      <ScrollView>
      <View style={BOX_SQUARES} >
        <TouchableOpacity style={BOX_SQUARE} onPress={()=>navigation.navigate("نقاطي")}>
          {/*points*/}
          <Image  source={points_box}  style={IMAGE_BOX} />
          <Text style={TEXT_BOX}>نقاطي</Text>


        </TouchableOpacity>
        <TouchableOpacity style={BOX_SQUARE} onPress={()=>navigation.navigate("ارباحي")} >
            {/*earnings*/}
          <Image  source={earnings_box}  style={IMAGE_BOX}  />
          <Text style={TEXT_BOX} >ارباحي</Text>


        </TouchableOpacity>
      </View>
      <View style={BOX_SQUARES} >
        <TouchableOpacity style={BOX_SQUARE} onPress={()=>navigation.navigate("conditions")} >
```

Figure 136: home 2

```
return (
  <Screen style={ROOT} preset="scroll">
    <ScrollView>
    <View style={BOX_SQUARES} >
      <TouchableOpacity style={BOX_SQUARE} onPress={()=>navigation.navigate("نقـاطي")}>
        {/*points*/}
        <Image  source={points_box}  style={IMAGE_BOX} />
        <Text style={TEXT_BOX}>نقـاطي</Text>


      </TouchableOpacity>
      <TouchableOpacity style={BOX_SQUARE} onPress={()=>navigation.navigate("ارباحي")} >
          {/*earnings*/}
        <Image  source={earnings_box}  style={IMAGE_BOX}  />
        <Text style={TEXT_BOX} >ارباحي</Text>


      </TouchableOpacity>
    </View>
    <View style={BOX_SQUARES} >
      <TouchableOpacity style={BOX_SQUARE} onPress={()=>navigation.navigate("conditions")} >
          {/*conditions*/}
        <Image  source={conditions_box}  style={IMAGE_BOX}  />
        <Text style={TEXT_BOX} >الشروط</Text>

      </TouchableOpacity>
      <TouchableOpacity style={BOX_SQUARE} onPress={()=>navigation.navigate("تـحديـاتي")} >
        {/*challenges*/}
        <Image  source={challenges_box}  style={IMAGE_BOX} />
        <Text style={TEXT_BOX} >تـحديـاتي</Text>


      </TouchableOpacity>
    </View>
    <View>
    <Video
source={video_index}
```

Figure 137: home 3

```
      </TouchableOpacity>
    </View>
    <View>
    <Video
 source={video_index}
 shouldPlay
 useNativeControls
 resizeMode="cover"

 style={{ width: 300, height: 200, alignSelf:'center',borderColor:'orange',margin:25,borderRadius:20,borderWidth:1,}}  />
    </View>
    <View style={NEWS}>
    <Ionicons name="megaphone" size={20} color={color.palette.angry} />
    <Text style={{color:color.palette.primary,fontSize:20,marginHorizontal:13,fontWeight:'bold'}} >اخبـار و جديـد الـسوق</Text>

    </View>
    <View>

    <Video
 source={video_index}
 shouldPlay
 useNativeControls
 resizeMode="cover"

 style={{ width: 300, height: 200, alignSelf:'center',borderColor:'orange',margin:25,borderRadius:20,borderWidth:1,}}  />
    </View>
    <View>

    <Video
 source={video_index}
 shouldPlay
 useNativeControls
 resizeMode="cover"

 style={{ width: 300, height: 200, alignSelf:'center',borderColor:'orange',margin:25,borderRadius:20,borderWidth:1,}}  />
    </View>
    </ScrollView>
```

Figure 138 : home 4

Dans cette partie on a appris un nouveau chose c’est l’importation, la lecture d’une vidéo .

```
  <Video
source={video_index}
shouldPlay
useNativeControls
resizeMode="cover"

style={{ width: 300, height: 200, alignSelf:'center',borderColor:'orange',margin:25,borderRadius:20,borderWidth:1,}}  />
```

Screens :

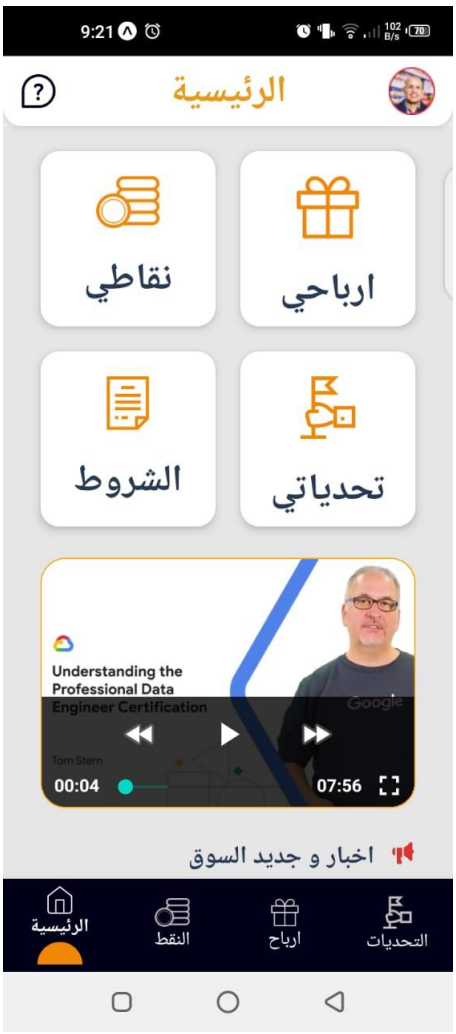


Figure 139: Home screen

- **La partie Conditions**

```
import React from "react"
import { observer } from "mobx-react-lite"
import { Dimensions,Image,ImageBackground,ImageStyle,TextStyle, TouchableOpacity, View, ViewStyle } from "react-native"
import { Button, Screen, Text } from "../../../components"
import { useNavigation } from "@react-navigation/native"
// import { useStores } from "../../models"
import { color } from "../../../theme"
import { Video } from 'expo-av';
import { Header } from 'react-native-elements';
import { back_ground } from "../..."
import { ScrollView } from "react-native-gesture-handler"
import {Linking} from 'react-native'

/*<Header
        containerStyle={{ backgroundColor: color.palette.white, borderBottomLeftRadius: 20, borderBottomRightRadius: 20, height: 62, }}
        leftComponent={<TouchableOpacity style={{ flexDirection: 'row', }} onPress={() => { navigation.goBack() } }><Image style={{ marginTop: 5
        centerComponent={{ text: 'MY TITLE', style: { color: color.palette.black, marginTop: 0.5, } }}
        rightComponent={<TouchableOpacity><Image style={{ marginBottom: 40, }} source={require("../../../images/profile.png")} /></TouchableOpac
        />

<TouchableOpacity style={{flexDirection:'row',}} onPress={()=>{}}><Image  style={{marginTop:5,marginHorizontal:10}} source={arrow_left}  /><Text
export const video_index=require("../../../videos/index.mp4")

const ROOT: ViewStyle = {
  backgroundColor: color.palette.offWhite,
  flex: 1,
}
const BOX : ViewStyle = {
  justifyContent:'center',


}
const CONDITIONS : ViewStyle={

  backgroundColor:color.palette.white,
  marginHorizontal:20,
  marginTop:20,
```

Figure 141: condition 1

```
}
const CONTACT_TEXT : ViewStyle = {
  marginHorizontal:12,
  marginVertical:10,

}
const users=[
  {id : "1",title : "الشرط الاول", describtion : "ر أدابيا يسكينج ألياية,سيت دو أيوسمود تيمبورأنكايديديونتيوت لابوري ات دولار ماجنا أليكيوا
 },
 {id : "2",title : "الشرط الثاني", describtion : "أدابيا يسكينج ألياية,سيت دو أيوسمود تيمبورأنكايديديونتيوت لابوري ات دولار ماجنا أليكيوا
},
{id : "3",title : "الشرط الثالث", describtion : "ر أدابيا يسكينج ألياية,سيت دو أيوسمود تيمبورأنكايديديونتيوت لابوري ات دولار ماجنا أليكيوا
},

]

const contacts_phones={tel:"0666666666",whatsapp:"https://www.whatsapp.com/?lang=fr"}
export const arrow_left = require("../../../images/arrow-left.png")
export const contactus_image = require("../../../images/contact-us.png")
export const whatsapp_icon = require("../../../images/whatsapp.png")
export const phone_icon = require("../../../images/tel.png")


export const ConditionsScreen = observer(function ConditionsScreen() {
  // Pull in one of our MST stores
  // const { someStore, anotherStore } = useStores()

  // Pull in navigation via hook
  const navigation = useNavigation()

  return (
    <Screen style={ROOT} preset="scroll">
```

Figure 140 : condition 2

```
    {
      users.map((cond,index)=>{
        return (

          <View style={CONDITION}>
          <View style={CONDITION_ROW}>
         <View  style={CERCLE_CONDITION}>
          <Text style={NUMBER_CONDITION}>{cond.id}</Text>

         </View>
         <Text style={TITLE_CONDITION}>{cond.title}</Text>
         </View>
         <Text style={DESCRIPTION_CONDITION}>{cond.describtion}</Text>

         </View>
        );

      })
    }

      </View>

  <Video
  source={video_index}
  shouldPlay
  useNativeControls
  resizeMode="cover"
  style={Video_Style} />
  <View style={Contact_us}>
  <ImageBackground  style={Image_Contact} source={contactus_image}>
  <View style={{}}>
  <View style={CONTACT_TEXT}>
          <Text style={{color:color.palette.orangeDarker,fontSize:20,fontWeight:'bold'}} >محتاج شي مساعدة ؟</Text>
          <Text  style={{color:color.palette.black,fontSize:20,fontWeight:'bold'}} >اتصل بنا </Text>
```

Figure 142: condition 3

```
resizeMode="cover"
style={Video_Style} />
<View style={Contact_us}>
<ImageBackground  style={Image_Contact} source={contactus_image}>
<View style={{}}>
<View style={CONTACT_TEXT}>
          <Text style={{color:color.palette.orangeDarker,fontSize:20,fontWeight:'bold'}} >محتاج شي مساعدة ؟</Text>
          <Text  style={{color:color.palette.black,fontSize:20,fontWeight:'bold'}} >اتصل بنا </Text>

      </View>
  <View style={CONTACTS}>
  <TouchableOpacity onPress={()=>{Linking.openURL(contacts_phones.whatsapp)}}>

        <View style={WHATSAPP}>
        <Text style={{color:color.palette.whatsapp,fontSize:15,fontWeight:'bold',marginLeft:30,marginRight:30,}}>واتساب</Text>
        <Image  source={whatsapp_icon} style={{alignSelf:'center',marginHorizontal:1,}}/>

        </View>
        </TouchableOpacity>

        <TouchableOpacity onPress={()=>{Linking.openURL("tel:${contacts_phones.tel}")}}>
        <View style={PHONE}>
        <Text style={{color:color.palette.black,fontSize:15,fontWeight:'bold',}}>0666666666</Text>
        <Image  source={phone_icon} style={{alignSelf:'center',marginHorizontal:5,}}/>
        </View>
        </TouchableOpacity>
  </View>

</View></ImageBackground>
```

Figure 143: condition 4

Screen :

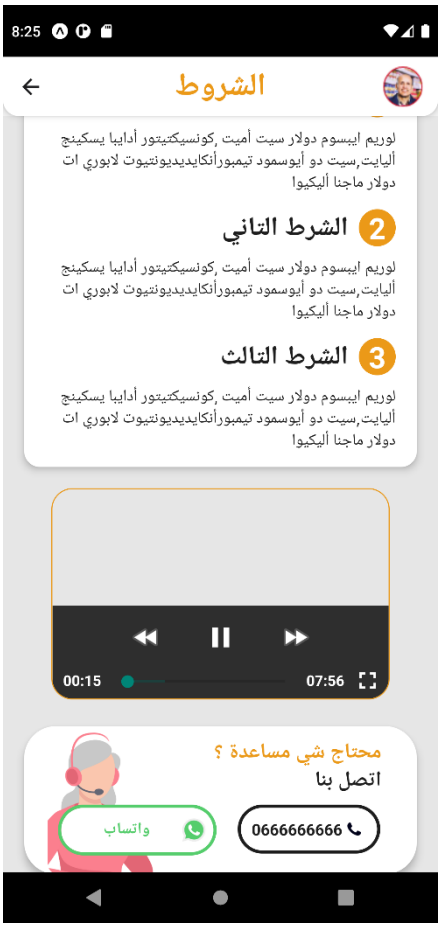


Figure 144: conditions

- **La partie Points**

```
import React from "react"
import { observer } from "mobx-react-lite"
import { View, ViewStyle } from "react-native"
import { Button, Header, Screen, Text } from "../../../components"
import { useNavigation } from "@react-navigation/native"
// import { useStores } from "../../models"
import { color } from "../../../theme"
import { ScrollView } from "react-native-gesture-handler"
import { Image } from "react-native-elements/dist/image/Image"
import Timeline from 'react-native-timeline-flatlist'
import { back_ground } from "../../"

const ROOT: ViewStyle = {
  backgroundColor: color.palette.offWhite,
  flex: 1,
}

const POINTS_CONTAINER :ViewStyle= {
  shadowColor: "#000",
shadowOffset: {
  width: 0,
  height: 2,
},
shadowOpacity: 0.25,
shadowRadius: 4.84,

elevation: 1,
margin:10,
//marginHorizontal:30,
borderRadius:20,
backgroundColor:color.palette.white,
alignSelf:'center',
flexDirection:'column',
width:370,
```

Figure 145: point 1

```
<Text style={{color:color.palette.orangeDarker ,fontWeight:'bold',}}>320</Text>
  </View>
  <View style={{flexDirection:'row-reverse',}}>
<Text style={{color:color.palette.orangeDarker,marginLeft:100}}>صور كوكا وربيج</Text>
  <Text style={{color:color.palette.black ,fontWeight:'bold'}}>نقطة </Text>
  </View>
        </View>
    );

  }
   const navigation = useNavigation()
   const data = [
    {title:"سجل النقط",icon:require("../../../images/history2.png")},
    { title : line3(),icon:require("../../../images/coke.png"),separator :true},
    { title : line3(),icon:require("../../../images/sprite.png"),separator :true},
    { title : line3(),icon:require("../../../images/coke.png"),separator :true},
    { title : line3(),icon:require("../../../images/sprite.png"),separator :true},
    { title : line3(),icon:require("../../../images/sprite.png"),separator :true},



  ]
  return (
    <Screen style={ROOT} preset="scroll"   >
      <ScrollView>
        <View style={POINTS_CONTAINER}>
          <View style={POINTS_TOTAL}>
            <Image  source={points} style={{width:51,height:40,}} />
            <Text style={{color: color.palette.black,fontSize:25,fontWeight:'bold',marginHorizontal:20,}}>مجموع النقط</Text>

          </View>
          <Text style={{fontSize:65,color:color.palette.orangeDarker,alignSelf:'flex-end',marginHorizontal:30,fontWeight:'bold',}}>3056</Text>
          <Text style={{color: color.palette.black,fontSize:25,fontWeight:'bold',marginHorizontal:30,marginBottom:20,}}>نقطة</Text>

          <View>
```

Figure 146: point 2

```
        <View style= {POINTS_JOURNAL}>
        <Timeline
        data={data}
        circleSize={30}
        circleStyle={{backgroundColor:color.palette.white}}
        innerCircle={'icon'}
        columnFormat={"single-column-right"}
        lineColor={color.palette.lighterGrey}

        style={{marginVertical:20,}}
        separator={true}
        separatorStyle={{borderColor:color.palette.lighterGrey,marginHorizontal:20,marginLeft:35,borderWidth:0.1,}}
        titleStyle={{alignSelf:'flex-end',}}

        />
        </View>

        </ScrollView>


      </Screen>
    )
  })
```

Figure 147: point 3

Dans cette partie on a appris un nouveau chose c'est l'importation d'un TimeLine pour afficher l'historique des points .

```
<Timeline
data={data}
circleSize={30}
circleStyle={{backgroundColor:color.palette.white}}
innerCircle={'icon'}
columnFormat={"single-column-right"}
lineColor={color.palette.lighterGrey}

style={{marginVertical:20,}}
separator={true}
separatorStyle={{borderColor:color.palette.lighterGrey,marginHorizontal:20,marginLeft:35,borderWidth:0.1,}}
titleStyle={{alignSelf:'flex-end',}}

/>
```

Screen :

Figure 148: Points screen 1

Figure 149: Points Screen 2

- **La partie Gains (Earnings)**

```
import React from "react"
import { observer } from "mobx-react-lite"
import { FlatList, I18nManager, Image, ImageStyle, StyleSheet, TextStyle, View, ViewStyle } from "react-native"
import { Screen, Text } from "../../../components"
import { useNavigation } from "@react-navigation/native"
// import { useStores } from "../../models"
import { color } from "../../../theme"
import { ScrollView } from "react-native-gesture-handler"
import { ProgressBar, Colors } from 'react-native-paper';
import GridView from 'react-native-easy-gridview';
import { Video } from 'expo-av';

export const earning_icon =require("../../../images/feather-gift.png");
export const earning_icon_red =require("../../../images/red-gift.png");


const ROOT: ViewStyle = {
  backgroundColor: color.palette.offWhite,
  flex: 1,
}
const View_Title : ViewStyle = {
    flexDirection:'row-reverse',
    //alignSelf:'flex-end',
    padding:20,
    paddingHorizontal:30,

}
const Title_Style : TextStyle = {
  color:color.palette.black,
  fontSize:20,
  fontWeight:'bold',
  margin:1,
  marginRight:10,
}
```

Figure 150: gain 1

```
 export const video_index=require("../../../videos/index.mp4")

export const EarningsScreen = observer(function EarningsScreen() {
  // Pull in one of our MST stores
  // const { someStore, anotherStore } = useStores()

  // Pull in navigation via hook
  const data    = [
    {title : "ساعة ذكية", image :require("../../../images/watch.png"),point_number:"320"},

    {title : "الة القهوة", image :require("../../../images/nesspresso.png"),point_number:"380"},
    {title : "مكبر الصوت", image :require("../../../images/speaker.png"),point_number:"340"},
    {title : "سماعات", image :require("../../../images/headset.png"),point_number:"320"},

  ]
  const grid1    = [
    {title : "هاتف ذكي ", image :require("../../../images/smart-phone.png"),point_number:"320"},
    {title : "سماعات", image :require("../../../images/headset.png"),point_number:"320"},

    {title : "الة القهوة", image :require("../../../images/nesspresso.png"),point_number:"380"},
    {title : "مكبر الصوت", image :require("../../../images/speaker.png"),point_number:"340"},
    {title : "هاتف ذكي ", image :require("../../../images/smart-phone.png"),point_number:"320"},
    {title : "هاتف ذكي ", image :require("../../../images/smart-phone.png"),point_number:"320"},
  ]
  const grid2    = [
    {title : "دراجة نارية ", image :require("../../../images/smart-phone.png"),point_number:"120"},
    {title : "دراجة ثلاثية", image :require("../../../images/smart-phone.png"),point_number:"220"},
    {title : "هاتف ذكي ", image :require("../../../images/smart-phone.png"),point_number:"200"},
    {title : "سماعات", image :require("../../../images/headset.png"),point_number:"20"},

    {title : "الة القهوة", image :require("../../../images/nesspresso.png"),point_number:"80"},
    {title : "مكبر الصوت", image :require("../../../images/speaker.png"),point_number:"100"},

  ]
```

Figure 151: gain 2

```
{
  data.map((box,index)=>{
    return (
     <View key={index} style={Box_Gift}>
     <Text style={Box_Title}>{box.title}</Text>
     <View style={BOX_Content}>

     <Image   style= {Box_Image} source={box.image} />
     <View style={Points_Column}>
       <Text style={{color:color.palette.orangeDarker,fontSize:16,fontWeight:'bold'}}>{box.point_number}</Text>
       <Text style={{color:color.palette.lightGrey,fontSize:13,}}>نقطة</Text>

       </View>


     </View>
     <ProgressBar  progress={box.point_number/500} style={{ width: 100, borderRadius: 20, }} color={color.palette.orangeDarker} accessibilityValue={undefin

    </View>
    );

   })
  }
```

Figure 154: gain 3

```
 <View>
 <FlatList
 style={{transform: [{ scaleX: -1 }],
}}
  data={grid1}
  keyExtractor={(item, index) => index}
  numColumns={3}
  renderItem={({item}) =>
  <View style={Box_Gift}>
  <Text style={Box_Title}>{item.title}</Text>
  <View style={BOX_Content}>

  <Image   style= {Box_Image}source={item.image}/>
  <View style={Points_Column}>
    <Text style={{color:color.palette.orangeDarker,fontSize:16,}}>{item.point_number}</Text>
    <Text style={{color:color.palette.lightGrey,fontSize:13,}}>نقطة</Text>

    </View>


  </View>
  <ProgressBar  progress={item.point_number / 500} style={{ width: 100, borderRadius: 20, }} color={color.palette.orangeDarker} accessibilityValue={undefined} focus

 </View>
  }
/>

 </View>
 <View style={View_Title} >
      <Image style={Image_Title} source={earning_icon_red} />
      <Text style={Title_Style}>هدايا اخرى</Text>
```

Figure 153: gain 4

```
</View>
{/*grid view 2  */}
<View>
<FlatList
style={{transform: [{ scaleX: -1 }],
}}
  data={grid2}
  keyExtractor={(item, index) => index}
  numColumns={3}
  renderItem={({item}) =>
  <View style={Box_Gift}>
  <Text style={Box_Title}>{item.title}</Text>
  <View style={BOX_Content}>

  <Image   style= {Box_Image}source={item.image}/>
  <View style={Points_Column}>
    <Text style={{color:color.palette.angry,fontSize:16,}}>{item.point_number}</Text>
    <Text style={{color:color.palette.lightGrey,fontSize:13,}}>نقطة</Text>

    </View>


  </View>
  <ProgressBar  progress={item.point_number / 500} style={{ width: 100, borderRadius: 20, }} color={color.palette.angry} accessibilityValue={undefined} focusable={u

 </View>
  }
/>

 </View>
 <Video
source={video_index}
shouldPlay
```

Figure 152: gain 5

Dans cette partie on a appris un nouveau chose c'est l'importation d'un GridView pour afficher les gains d'un épicier .

```
{/*grid view 1  */}
<View>
<FlatList
style={{transform: [{ scaleX: -1 }],
}}
 data={grid1}
 keyExtractor={(item, index) => index}
 numColumns={3}
 renderItem={({item}) =>
 <View style={Box_Gift}>
 <Text style={Box_Title}>{item.title}</Text>
 <View style={BOX_Content}>

 <Image   style= {Box_Image}source={item.image}/>
 <View style={Points_Column}>
   <Text style={{color:color.palette.orangeDarker,fontSize:16,}}>{item.point_number}</Text>
   <Text style={{color:color.palette.lightGrey,fontSize:13,}}>نقطة</Text>

   </View>


 </View>
 <ProgressBar  progress={item.point_number / 500} style={{ width: 100, borderRadius: 20, }} color={color.palette.orangeDarker} accessibilityValue={undefined} focus

</View>
 }
/>
```

Screens :

Figure 156: gain screen 1

Figure 155: gain screen 2

- **La partie Challenges**

Cette partie est un peu complique parce que elle composé de plusieurs screen à naviguer.

L'architecture de la partie challenges :

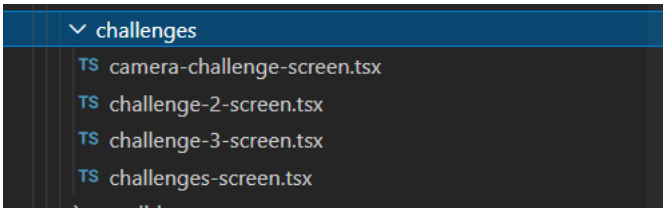


Figure 157: architecture des challenges

```
import React from "react"
import { observer } from "mobx-react-lite"
import { ImageBackground, ImageStyle, View, ViewStyle } from "react-native"
import { Screen, Text } from "../../../components"
import { useNavigation } from "@react-navigation/native"
// import { useStores } from "../../models"
import { color } from "../../../theme"
import { ScrollView, TouchableOpacity } from "react-native-gesture-handler"
import { Image } from "react-native-elements/dist/image/Image"
import { ProgressBar, Colors } from 'react-native-paper';
const ROOT: ViewStyle = {
  backgroundColor: color.palette.offWhite,
  flex: 1,
}
const Challenge_Box : ViewStyle = {
  borderRadius:15,
  shadowColor: "#000",
shadowOffset: {
  width: 0,
  height: 2,
},
shadowOpacity: 0.25,
shadowRadius: 4.84,
backgroundColor:color.palette.white,
elevation: 5,
marginTop:20,
margin:10,
flexDirection:"row",
justifyContent:'space-between',
}
const Image_Back_Challenge: ImageStyle = {
  width:180,height:128
}
const Image_inside : ImageStyle={
  width:109,height:34,margin:15,
}
const Challenge_Content : ViewStyle = {
```

Figure 158: challenge 1.1

```
const title_challenge : ViewStyle = {
  flexDirection : 'row-reverse',
  marginTop:25,
  marginLeft:20,

}
const image_challenge_mini: ImageStyle ={
  height:15,width:50,margin:8,
}
const data = [{id:1,name:"hello 1"},{id:2,name:"hello 2"},{id:3,name:"hello 3"}]
const challenges_data= [
  {id : 1, title:"تحدي كوكاكولا",description : "وريم ايبسوم دولار سيت أميت ",point_number:320,},
  {id : 2, title:"تحدي كوكاكولا",description : "وريم ايبسوم دولار سيت أميت ",point_number:100,},
  {id : 3, title:"تحدي كوكاكولا",description : "وريم ايبسوم دولار سيت أميت ",point_number:450,},
  {id : 4, title:"تحدي كوكاكولا",description : "وريم ايبسوم دولار سيت أميت ",point_number:350,},
  {id : 5, title:"تحدي كوكاكولا",description : "وريم ايبسوم دولار سيت أميت ",point_number:40,},
  {id : 6, title:"تحدي كوكاكولا",description : "وريم ايبسوم دولار سيت أميت ",point_number:230,},
  {id : 7, title:"تحدي كوكاكولا",description : "وريم ايبسوم دولار سيت أميت ",point_number:30,},


]
export const ChallengesScreen = observer(function ChallengesScreen() {
  // Pull in one of our MST stores
  // const { someStore, anotherStore } = useStores()

  // Pull in navigation via hook
  const navigation = useNavigation()
  function challenge1 ( ){
    return (
      <TouchableOpacity style={Challenge_Box}>

      <ImageBackground source = {require("../../../images/intersection-second.png")} style={Image_Back_Challenge}>

        <Image source={require("../../../images/coke-mono.png")} style={Image_inside}/>
        <TouchableOpacity>
        <Image  style={{height:15,width:15,margin:20,}} source={require("../../../images/white-arrow.png")}  />
```

Figure 160: challenge 1.2

```
return (
  <Screen style={ROOT} preset="scroll">
    <ScrollView>
        {/*challenge 1 */}


      {/*challenge 2 */}

      <View >
        {
          challenges_data.map((data,index)=>{
            if (data.id%2!==0) {

              return (
                <TouchableOpacity style={Challenge_Box} onPress={()=>navigation.navigate("challenge_2")}>

    <ImageBackground source = {require("../../../images/intersection-second.png")} style={Image_Back_Challenge}>

      <Image source={require("../../../images/coke-mono.png")} style={Image_inside}/>
      <TouchableOpacity>
      <Image  style={{height:15,width:15,margin:20,}} source={require("../../../images/white-arrow.png")}  />
      </TouchableOpacity>
      </ImageBackground>
      <View style={Challenge_Content}>
        <View style={title_challenge}>
          <Image  source ={require("../../../images/paper.png") } style={{height:22,width:25,}}/>
        <Text style={{color:color.palette.black,fontSize:20,fontWeight:'bold',marginRight:10,}}>{data.title}</Text>

        </View>
        <Text style={{color:color.palette.lightGrey,fontSize:15,marginRight:55,}}>{data.description}</Text>

        <Text style={{color:color.palette.orange,fontSize:15,fontWeight:'bold', marginRight:55,}}> {data.point_number}/500</Text>

        <ProgressBar  progress={data.point_number / 500} color={color.palette.orangeDarker} style={{ borderRadius: 20, width: 160, }} accessibilit
```

Figure 159: challenge 1.3

```
</TouchableOpacity>
        );
          }
          else {
            return (
              <TouchableOpacity style={Challenge_Box} onPress={()=>navigation.navigate("challenge_2")} >

              <ImageBackground source = {require("../../../images/intersection.png")} style={Image_Back_Challenge}>

                <View style={{flexDirection:'column'}}>
                  <View style={{flexDirection:'row'}}>
                    <Image source={require("../../../images/eau.png")}  style={image_challenge_mini}/>
                    <Image source={require("../../../images/sidi-ali.png")}  style={image_challenge_mini}/>

                  </View>
                  <View style={{flexDirection:'row'}}>
                  <Image source={require("../../../images/sprite.png")}  style={image_challenge_mini}/>
                    <Image source={require("../../../images/hawai.png")}  style={image_challenge_mini}/>
                  </View>

                  </View>
                <TouchableOpacity>
                <Image  style={{height:15,width:15,margin:20,}} source={require("../../../images/red-arrow.png")}  />
                </TouchableOpacity>
                </ImageBackground>
                <View style={Challenge_Content}>
                  <View style={title_challenge}>
                    <Image  source ={require("../../../images/paper.png") } style={{height:22,width:25,}}/>
                  <Text style={{color:color.palette.black,fontSize:20,fontWeight:'bold',marginRight:10,}}> {data.title}</Text>

                  </View>
                  <Text style={{color:color.palette.lightGrey,fontSize:15,marginRight:55,}}>{data.description} </Text>

                  <Text style={{color:color.palette.orange,fontSize:15,fontWeight:'bold', marginRight:55,}}> {data.point_number}/500</Text>

                  <ProgressBar  progress={data.point_number / 500} color={color.palette.orangeDarker} style={{ borderRadius: 20, width: 160, }}
```

Figure 162: challenge 1.4

Screen :

Figure 161: challenge 1

```
import React from "react"
import { observer } from "mobx-react-lite"
import { Image, ImageBackground, ImageStyle, TextStyle, TouchableOpacity, View, ViewStyle } from "react-native"
import { Screen, Text } from "../../../components"
import { useNavigation } from "@react-navigation/native"
// import { useStores } from "../../models"
import { color } from "../../../theme"
import { ScrollView } from "react-native-gesture-handler"
import { ProgressBar } from "react-native-paper"

const ROOT: ViewStyle = {

  backgroundColor: color.palette.offWhite,
  flex: 1,
}
const Challenge_Box : ViewStyle = {
  borderRadius:15,
  shadowColor: "#000",
shadowOffset: {
  width: 0,
  height: 2,
},
shadowOpacity: 0.25,
shadowRadius: 4.84,
backgroundColor:color.palette.white,
elevation: 5,
marginTop:20,
margin:10,
flexDirection:"row",
justifyContent:'space-between',
}
const Image_Back_Challenge: ImageStyle = {
  width:180,height:128
}
const Image_inside : ImageStyle={
  width:109,height:34,margin:15,
}
```

Figure 163: challenge 2.1

```
const factures=[

  {title:"صور الفاكتورا ديال كوكاكولا",image:require("../../../images/camera2.png"),point_facture:320,},
  {title:"كيفاش تصنف السلعة",image:require("../../../images/play2.png"),point_facture:213,},
  {title:"جاوب على اسئلة المعاركة",image:require("../../../images/qa2.png"),point_facture:225,circle:require("../../../images/check-circle.png")},
  {title:"جاوب على اسئلة المعاركة",image:require("../../../images/qa2.png"),point_facture:225,circle:require("../../../images/check-circle.png")},
  {title:"جاوب على اسئلة المعاركة",image:require("../../../images/qa2.png"),point_facture:225,circle:require("../../../images/check-circle.png")},

]

export const Challenge2Screen = observer(function Challenge2Screen() {
  // Pull in one of our MST stores
  // const { someStore, anotherStore } = useStores()

  // Pull in navigation via hook
  const navigation = useNavigation()
  function percent (a,b){
    const perc=(a/b)*100
    return perc.toFixed(0)
  }
  return (
    <Screen style={ROOT} preset="scroll">
      <ScrollView>
      <TouchableOpacity style={Challenge_Box}>

    <ImageBackground source = {require("../../../images/intersection-second.png")} style={Image_Back_Challenge}>

      <Image source={require("../../../images/coke-mono.png")} style={Image_inside}/>
      <TouchableOpacity>
      <Image  style={{height:15,width:15,margin:20,}} source={require("../../../images/white-arrow.png")}  />
      </TouchableOpacity>
      </ImageBackground>
      <View style={Challenge_Content}>
        <View style={title_challenge}>
```

Figure 164: challenge 2.2

```
return (
  <Screen style={ROOT} preset="scroll">
     <ScrollView>
     <TouchableOpacity style={Challenge_Box}>

    <ImageBackground source = {require("../../../images/intersection-second.png")} style={Image_Back_Challenge}>

     <Image source={require("../../../images/coke-mono.png")} style={Image_inside}/>
     <TouchableOpacity>
     <Image  style={{height:15,width:15,margin:20,}} source={require("../../../images/white-arrow.png")}  />
     </TouchableOpacity>
     </ImageBackground>
     <View style={Challenge_Content}>
       <View style={title_challenge}>
         <Image  source ={require("../../../images/paper.png") } style={{height:22,width:25,}}/>
       <Text style={{color:color.palette.black,fontSize:20,fontWeight:'bold',marginRight:10,}}>تحدي كوكاكولا</Text>

       </View>
       <Text style={{color:color.palette.lightGrey,fontSize:15,marginRight:55,}}>لوريم ايبسوم دولار سيت أميت</Text>
       <View style={{flexDirection:'row',}}>
       <Text style={{color:color.palette.orange,fontSize:15,fontWeight:'bold',marginRight:70, }}> {percent(224,500)}%</Text>

       <Text style={{color:color.palette.lighterGrey,fontSize:15,fontWeight:'bold', }}>224/500 </Text>

         </View>
       <ProgressBar  progress={320 / 500} color={color.palette.orangeDarker} style={{ borderRadius: 20, width: 160, }} accessibilityValue={undefined} focusable={undefi

     </View>
   </TouchableOpacity>
   {/*factures */}
   {/*
```

Figure 166: challenge 2.3

```
   {
    factures.map((factures,index)=>{
      return (
        <TouchableOpacity style={Facture_Box} onPress={()=>navigation.navigate("challenge_3")}>
        <View>
        <View style={Title_Facture}>
          <Text style={Title_Text}>{factures.title}</Text>
          <Image  style={Title_Image} source={factures.image}/>
        </View>
        <View style={Point_Facture}>
          <Text style={{color:color.palette.orangeDarker,fontSize:20,fontWeight:'bold'}} > {factures.point_facture}</Text>
          <Text style={{color:color.palette.black,fontSize:17,}} > نقطة</Text>

        </View>
        </View>
        <Image source={factures.circle} style={Image_Circle}/>


       </TouchableOpacity>
       );

    })
   }

    </ScrollView>
  </Screen>
```

Figure 165: challenge 2.4

Screen :

Figure 167: challenge 2

```
import React, { useEffect, useRef, useState } from "react"
import { observer } from "mobx-react-lite"
import { Alert, Image, ImageBackground, ImageStyle, Platform, TextStyle, TouchableOpacity, View, ViewStyle } from "react-native"
import { Button, Screen, Text } from "../../../components"
import { useNavigation } from "@react-navigation/native"
// import { useStores } from "../../models"
import { color } from "../../../theme"
import { ScrollView } from "react-native-gesture-handler"
import { ProgressBar } from "react-native-paper"
import {Camera} from 'expo-camera'
import { Video } from 'expo-av';
import * as ImagePicker from 'expo-image-picker';
import RNGestureHandlerButton from "react-native-gesture-handler/lib/typescript/components/GestureHandlerButton"


const ROOT: ViewStyle = {
  backgroundColor: color.palette.offWhite,
  flex: 1,
}
let camera: Camera

const Challenge_Box : ViewStyle = {

flexDirection:"row",
justifyContent:'space-between',
backgroundColor:color.palette.white,
}
const Image_Back_Challenge: ImageStyle = {
  width:180,height:128
}
const Image_inside : ImageStyle={
  width:109,height:34,margin:15,
}
const Challenge_Content : ViewStyle = {
  flexDirection:"column",
}
const title_challenge : ViewStyle = {
```

Figure 169: challenges 3.1

```
export const Challenge3Screen = observer(function Challenge3Screen() {
  // Pull in one of our MST stores
  // const { someStore, anotherStore } = useStores()

  // Pull in navigation via hook
  const navigation = useNavigation()
 const [image,setImage]=useState(null);
 useEffect(()=>{
   (async ()=>{
     if (Platform.OS !== 'web'){
       const {status}=await ImagePicker.requestMediaLibraryPermissionsAsync();
       if (status!=='granted'){
          Alert.alert("sorry,we need camera roll permessions to make this wokr !")
       }

     }
   })
 })
 const pickImage= async()=>{
   let result = await ImagePicker.launchImageLibraryAsync({
     mediaTypes :ImagePicker.MediaTypeOptions.All,
     allowsEditing :true,
     aspect : [4,3],
     quality :1,
   });
   if (!result.cancelled) {
    setImage(result.uri);
  }
 }
```

Figure 168: challenges 3.2

```
  return (
    <Screen style={ROOT} preset="scroll">
      <ScrollView>

      |
      <View style={Challenge_Box}>

<ImageBackground source = {require("../../../images/blue-intersection.png")} style={Image_Back_Challenge}>
<View style={{flexDirection:"column",marginHorizontal:20,}}>
  <Text style={{color:color.palette.white,fontSize:55,} >320</Text>
  <Text style={{color:color.palette.white,fontSize:30,alignSelf:"flex-start",marginHorizontal:30,}} >نقطة</Text>
</View>
  </ImageBackground>
  <View style={Challenge_Content}>
    <View style={title_challenge}>
     <Image  source ={require("../../../images/camera.png") } style={{height:22,width:25,}}/>
    <Text style={{color:color.palette.black,fontSize:20,fontWeight:'bold',marginRight:10,}}>تحدي كوكاكولا</Text>

    </View>
    <Text style={{color:color.palette.lightGrey,fontSize:15,marginRight:55,}}>لوريم ايبسوم دولار سيت أميت</Text>

    <Text style={{color:color.palette.orange,fontSize:15,fontWeight:'bold', marginRight:55,}}> 320</Text>

    <ProgressBar  progress={320 / 500} color={color.palette.orangeDarker} style={{ borderRadius: 20, width: 160, }} accessibilityValue={undefined} focusable={undefined}/>

  </View>

</View>
<View>

<View style={challenge_description}>
<Text style={challenge_qts}>؟شنو هو هاد التحدي</Text>
<Text style={text_description}>لوريم ايبسوم دولار سيت أميت ,كونسيكتيتور أدايبا يسكينج أليايت,سيت دو أيوسمود تيمبور
```

Figure 170: challenges 3.3

```
<View style={challenge_description}>
<Text style={challenge_qts}>؟شنو هو هاد التحدي</Text>
<Text style={text_description}>لوريم ايبسوم دولار سيت أميت ,كونسيكتيتور أدايبا يسكينج أليايت,سيت دو أيوسمود تيمبور

أنكايديديونتيوت لابوري ات دولار ماجنا أليكيوا . يوت انيم أد مينيم فينايم,كيواس نوستريد </Text>
</View>

<View style={Camera_Column} >
<TouchableOpacity style={{margin:10,}} onPress={()=>navigation.navigate("challenge_camera")}>
<ImageBackground  source = {require("../../../images/take-photo.png")} style={take_photo}>
<Text style={take_photo_text} >التقط الصورة</Text>

</ImageBackground>
</TouchableOpacity>

<TouchableOpacity  style={{margin:10,}} onPress={pickImage}>
{image && <Image source={{ uri : image } } style={{ width: 200, height: 200 }} />}

<ImageBackground  source = {require("../../../images/upload-photo.png")} style={pick_picture}>
<Text style={pick_picture_text} >حمل الصورة ديالك</Text>

</ImageBackground>
</TouchableOpacity>

<TouchableOpacity style={send_photo}>
      <Image source={require("../../../images/send.png")}  style={send_photo_image} />
      <Text style={send_text} >ارسال الصورة</Text>

</TouchableOpacity>


</View>

</View>
<View>
```

Figure 171: challenges 3.4

Dans cette partie on a appris un nouveau chose c'est comment choisir une image de gallérie et l'importer via l'application **RED** .

```
const pickImage= async()=>{
  let result = await ImagePicker.launchImageLibraryAsync({
    mediaTypes :ImagePicker.MediaTypeOptions.All,
    allowsEditing :true,
    aspect : [4,3],
    quality :1,
  });
  if (!result.cancelled) {
   setImage(result.uri);
  }
}
```

Figure 172: pick image

Screens :

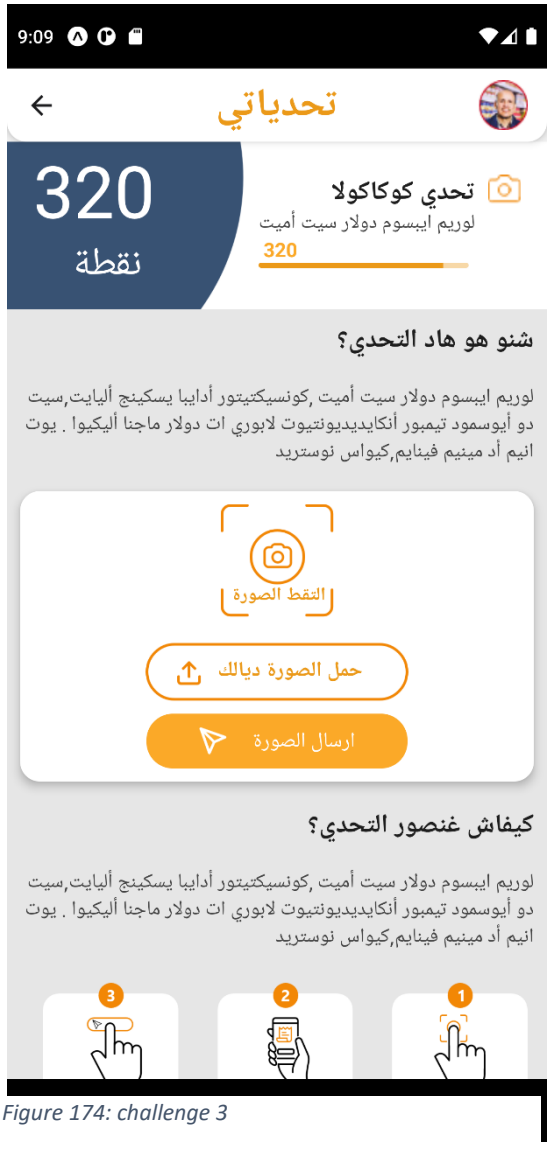


Figure 174: challenge 3

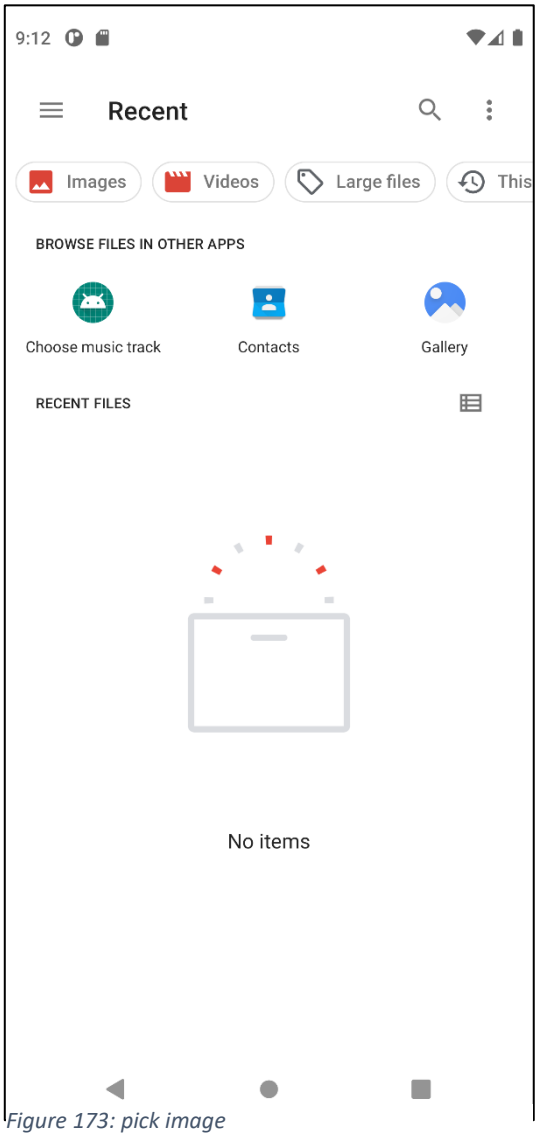


Figure 173: pick image

```tsx
import React, { useEffect, useRef, useState } from "react"
import { observer } from "mobx-react-lite"
import { Image, Text, TouchableOpacity, View, ViewStyle } from "react-native"
import { Screen } from "../../../components"
// import { useNavigation } from "@react-navigation/native"
// import { useStores } from "../../models"
import { color } from "../../../theme"
import {Camera } from "expo-camera"
import Ionicons from "@expo/vector-icons/build/Ionicons"
import * as FileSystem from 'expo-file-system';
import * as MediaLibrary from "expo-media-library";
import * as Permissions from "expo-permissions";
//https://github.com/adityakmr7/react-native-expo-camera-tutorial/blob/master/App.tsx
const ROOT: ViewStyle = {
  backgroundColor: "transparent",
  flex: 1,
}
const Camera_View : ViewStyle={
  flex:1,
  flexDirection:'column-reverse',

}
const image_swipe : ViewStyle={
  //height:40,
  backgroundColor:"transparent",
  alignItems:'center',
  flexDirection:'row',
  justifyContent:'center',



}
export const CameraChallengeScreen = observer(function CameraChallengeScreen() {
  // Pull in one of our MST stores
  // const { someStore, anotherStore } = useStores()

  // Pull in navigation via hook
```

Figure 176: camera 1

```tsx
export const CameraChallengeScreen = observer(function CameraChallengeScreen() {
  // Pull in one of our MST stores
  // const { someStore, anotherStore } = useStores()

  // Pull in navigation via hook
  // const navigation = useNavigation()
  const USER_PHOTO_DIR = FileSystem.documentDirectory + 'photos';
  const gallery ="C:\Users\ahafdi\Desktop\react native\gear9_projects\mobile_projects\RED_proj\RED_proj\app\photos";
  const [image_Preview,setimage_Preview]=useState("image")
  const [hashPermission,setHashPermission]=useState(null);
  const [type,setType]=useState(Camera.Constants.Type.back);
  const ref = useRef(null)

  useEffect(() => {
    (async () => {
      const { status } = await Camera.requestPermissionsAsync();
      setHashPermission(status === 'granted');
    })();
  }, []);

  if (hashPermission === null) {
    return <View />;
  }
  if (hashPermission === false) {
    return <Text>No access to camera</Text>;
  }
  const _takePhoto = async () => {
      if (ref.current) {
        const imageName = `${Date.now()}.jpg`;
    const photoSource = `${USER_PHOTO_DIR}/${imageName}`;
        const options = { quality: 0.5, base64: true };
        const photo = await ref.current.takePictureAsync(options)
        await FileSystem.copyAsync({
          from: photo.uri,
          to: photoSource,
        })
```

Figure 175: camera 2

```
  const handleSave = async (photo: string) => {
    const { status } = await Permissions.askAsync(Permissions.CAMERA_ROLL);
    if (status === "granted") {
      const assert = await MediaLibrary.createAssetAsync(photo);
      await MediaLibrary.createAlbumAsync("red_project", assert);
    } else {
      console.log("Oh You Missed to give permission");
    }
  };

  return (
    <Screen style={ROOT} preset="scroll">

<Camera  style={Camera_View} type={type}  ref={ref} >
        <View style={image_swipe}  >

        <TouchableOpacity onPress={()=>console.log("images")}>

                <Ionicons name="images-outline" size={30} color={color.palette.white} style={{alignSelf:"center",}}  />

                </TouchableOpacity>
                <TouchableOpacity onPress={_takePhoto}>

        <Ionicons name="ellipse-outline" size={80} color={color.palette.white} style={{alignSelf:"center",marginHorizontal:100,}}  />
</TouchableOpacity>
        <TouchableOpacity onPress={()=>{setType(
          type===Camera.Constants.Type.back? Camera.Constants.Type.front : Camera.Constants.Type.back
        )}}
        >
          <Ionicons name="camera-reverse-outline" size={30} color={color.palette.white} style={{alignSelf:"center",}}  />



        </TouchableOpacity>
```

Figure 177:camera 3

```
<Camera  style={Camera_View} type={type}  ref={ref} >
        <View style={image_swipe}  >

        <TouchableOpacity onPress={()=>console.log("images")}>

                <Ionicons name="images-outline" size={30} color={color.palette.white} style={{alignSelf:"center",}}  />

                </TouchableOpacity>
                <TouchableOpacity onPress={_takePhoto}>

        <Ionicons name="ellipse-outline" size={80} color={color.palette.white} style={{alignSelf:"center",marginHorizontal:100,}}  />
</TouchableOpacity>
        <TouchableOpacity onPress={()=>{setType(
          type===Camera.Constants.Type.back? Camera.Constants.Type.front : Camera.Constants.Type.back
        )}}
        >
          <Ionicons name="camera-reverse-outline" size={30} color={color.palette.white} style={{alignSelf:"center",}}  />



        </TouchableOpacity>
          </View>
          <View>
            <Image  style={{height : 100,width:100,}} source ={{uri : image_Preview}}/>
          </View>

      </Camera>




    </Screen>
```

Figure 178: camera 4

Dans cette partie on a appris un nouveau chose c'est comment créer une camera ,prendre une photo et la stocker dans la gallérie du téléphone .

```
<Camera  style={Camera_View} type={type}  ref={ref} >
        <View style={image_swipe}  >

        <TouchableOpacity onPress={()=>console.log("images")}>

                <Ionicons name="images-outline" size={30} color={color.palette.white} style={{alignSelf:"center",}}  />

                </TouchableOpacity>
                <TouchableOpacity onPress={_takePhoto}>

        <Ionicons name="ellipse-outline" size={80} color={color.palette.white} style={{alignSelf:"center",marginHorizontal:100,}}  />
</TouchableOpacity>
        <TouchableOpacity onPress={()=>{setType(
          type===Camera.Constants.Type.back? Camera.Constants.Type.front : Camera.Constants.Type.back
        )}}
        >
          <Ionicons name="camera-reverse-outline" size={30} color={color.palette.white} style={{alignSelf:"center",}}  />



        </TouchableOpacity>
          </View>
          <View>
            <Image  style={{height : 100,width:100,}} source ={{uri : image_Preview}}/>
          </View>

      </Camera>




    </Screen>
```

Screens :

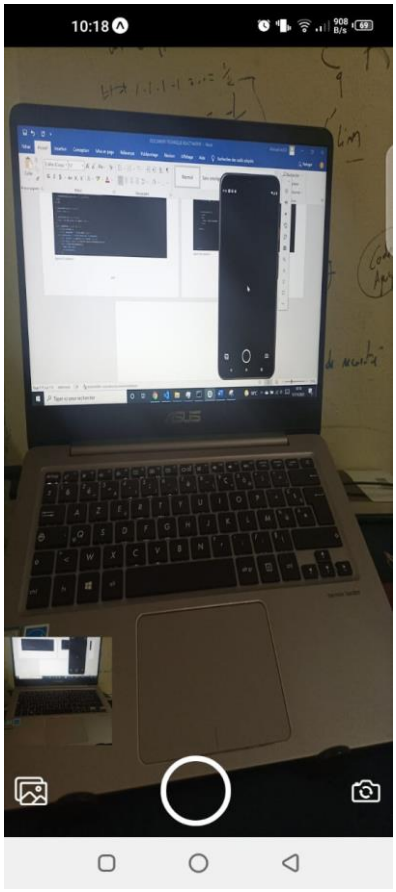


Figure 179: prendre une photo

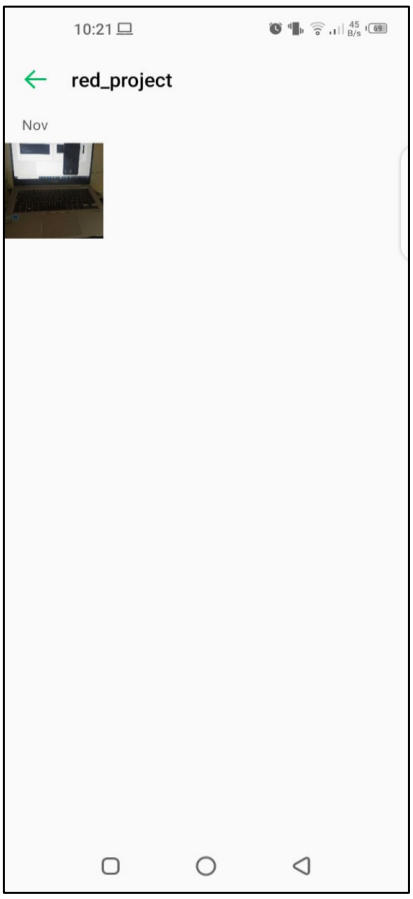


Figure 180: stocker la photo dans la gallérie

- **La partie Profile :**

```
import React, { useState } from "react"
import { observer } from "mobx-react-lite"
import { Image, ImageBackground, ImageStyle, TextStyle, View, ViewStyle } from "react-native"
import { Button, Screen, Text } from "../../../components"
import { useNavigation } from "@react-navigation/native"
// import { useStores } from "../../models"
import { color } from "../../../theme"
import Iframe from 'react-iframe';
import { TouchableOpacity } from "react-native-gesture-handler"
import { Camera } from "expo-camera"

const ROOT: ViewStyle = {
  backgroundColor: color.palette.white,
  position:'absolute',
  top:45,
  right: 0,

  //paddingHorizontal:30,
  //marginHorizontal:10,
  width:410,
  height:600,
  borderBottomLeftRadius:20,
  borderBottomRightRadius:20,
  shadowColor: "#000",
shadowOffset: {
  width: 0,
  height: 2,
},
shadowOpacity: 0.25,
shadowRadius: 4.84,

elevation: 5,
alignItems:'center',
```

Figure 181: profile 1

```
export const profile_image=require("../../../images/profile3.png")
export const camera_image=require("../../../images/profile-camera.png")
export const QR_code_border=require("../../../images/qr-border3.png")
export const setting_image=require("../../../images/settings.png")
export const logout_image=require("../../../images/logout.png")

export const ProfileScreen = observer(function ProfileScreen() {
  // Pull in one of our MST stores
  // const { someStore, anotherStore } = useStores()

  // Pull in navigation via hook
  const navigation = useNavigation()
  const [ShowProfile,setShowProfile]=useState(true);

  function  profile_view () {
    return(
      <View style={{width: 200, height: 300, backgroundColor:'red', position:'absolute',  top:80,borderRadius:20,}}><Text>hello</Text>


      </View>
    );
  }

  return (
    <View style={ROOT} >
      <View style={{marginTop:10,marginBottom:5,alignItems:'center'}}>
        {/*profile photo*/}
      <Image source ={profile_image} style={image_profile_style} />
      <Image style={camera_image_style}  source = {camera_image}/>

      </View>
      <Text style={Profile_Name}>حميد السعدي</Text>

      <View>
        {/* QR CODE */}
        <ImageBackground  style={qr_code_border_style} source ={QR_code_border} >
```

Figure 182: profile 2

```
return (
  <View style={ROOT} >
    <View style={{marginTop:10,marginBottom:5,alignItems:'center'}}>
      {/*profile photo*/}
    <Image source ={profile_image} style={image_profile_style} />
    <Image style={camera_image_style}  source = {camera_image}/>

    </View>
    <Text style={Profile_Name}>حميد السعدي</Text>

    <View>
      {/* QR CODE */}
      <ImageBackground  style={qr_code_border_style} source ={QR_code_border} >

      </ImageBackground>
       <Text style={qr_id}>AA3D23R-B</Text>

    </View>
    <View style={{margin:15,}}>
      {/*settings */}
      <View style={settings_border}>

      <Image  source={setting_image}/>
      <Text style={settings_text}>تغيير الاعدادات</Text>
      </View>
    </View>
    <TouchableOpacity onPress={()=>navigation.navigate("login")}>
      {/*log out  */}
      <View style={logout_border}>
        <Image source={logout_image} style={{height:15,width:17,marginTop:5,}}/>
        <Text style={logout_text}>تسجيل الخروج </Text>
      </View>
    </TouchableOpacity>
```

Figure 184: profile 3

Screens :

Figure 183: profile screen

Chapitre 31: Lecture des PDF & système de téléchargement

Introduction :

PDF Reader : est un composant de vue PDF natif réactif , prise en charge multiplateforme .

Caractéristiques :

- **lire un PDF à partir d'une URL, d'un blob, d'un fichier local ou d'un actif et peut le mettre en cache.**
- **afficher horizontalement ou verticalement**
- **glisser et zoomer**
- **appuyez deux fois pour zoomer**
- **soutien pdf protégé par mot de passe**
- **accéder à une page spécifique du pdf**

Architecture de projet :

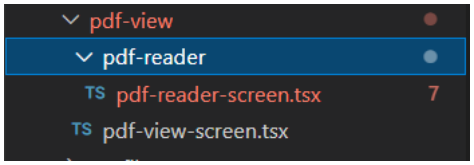


Figure 185: PDF Architecture

```tsx
import React from "react"
import { observer } from "mobx-react-lite"
import { TextStyle, ViewStyle } from "react-native"
import { Button, Screen, Text } from "../../components"
import { useNavigation } from "@react-navigation/native"
// import { useStores } from "../../models"
import { color } from "../../theme"
import { SafeAreaView } from "react-native-safe-area-context"
import metrics from "../../theme/metrics"
import Share from "react-native-share";

import Icon from 'react-native-vector-icons/FontAwesome';

export const PdfViewScreen = observer(function PdfViewScreen() {
  // Pull in one of our MST stores
  // const { someStore, anotherStore } = useStores()

  // Pull in navigation via hook
  const navigation = useNavigation()
  const url = "https://awesome.contents.com/";
  const title = "Awesome Contents";
  const message = "Please check this out.";
  const options = {
   title,
   url,
   message,
  };
```

Figure 186 : PDF view 1

```
  return (
    <Screen style={ROOT} preset="scroll">
      <SafeAreaView>
      <Text preset="header" text="PDF VIEW" style={{ alignSelf:'center',margin: metrics.heightPercentageToDP(2)}} />

      <Button text="SEE PDF FILE" style={Button_Pdf} textStyle={Button_Text} onPress={()=>navigation.navigate("pdf_reader")}/>
      <Button text="DOWNLOAD PDF FILE" style={Button_Pdf} textStyle={Button_Text}/>
      <Button
        onPress={async () => {
          await share();
        }}
        text="share"
      />
            <Icon
          name='share'
         // type='FontAwesome'
          size={60}
          color='orange'
          onPress={async () => { await share() } }   />
      </SafeAreaView>
    </Screen>
  )
})
const ROOT: ViewStyle = {
  backgroundColor: color.palette.black,
  flex: 1,
  alignItems:'center',
  justifyContent:'center',
}
```

Figure 187 : PDF view 2

```
import React, { useEffect } from "react"
import { observer } from "mobx-react-lite"
import { Alert, Dimensions, PermissionsAndroid, Platform, View, ViewStyle } from "react-native"
import { Screen, Text } from "../../../components"
import { useNavigation } from "@react-navigation/native"
// import { useStores } from "../../models"
import { color, spacing } from "../../../theme"
import Pdf from "react-native-pdf"
import Share from 'react-native-share';
import Icon from 'react-native-vector-icons/FontAwesome';
import metrics from "../../../theme/metrics"
const source = { uri: 'http://samples.leanpub.com/thereactnativebook-sample.pdf' };

import RNFetchBlob from 'rn-fetch-blob';

export const PdfReaderScreen = observer(function PdfReaderScreen() {
  // Pull in one of our MST stores
  // const { someStore, anotherStore } = useStores()

  // Pull in navigation via hook
  const navigation = useNavigation()
  const url = source.uri;
   const title = "Awesome Contents";
   const message = "Please check this out.";
   const options = {
    title,
    url,
    message,
  };
  const share = async (customOptions = options) => {
    try {
```

Figure 188: PDF reader 1

```
const download_pdf = async ()=>{
    try {
        const response = await fetch("http://samples.leanpub.com/thereactnativebook-sample.pdf");
        console.log(response);
    }catch(error){
      console.error(error)
    }
}
useEffect(()=>{
  //download_pdf();
});

const  DownloadHistory =async ()=> {
 const { config, fs } = RNFetchBlob;
 let PictureDir = fs.dirs.PictureDir;
 let date = new Date();
 let options = {
  fileCache: true,
  addAndroidDownloads: {
    //Related to the Android only
    useDownloadManager: true,
    notification: true,
    path:
      PictureDir +
      '/Report_Download' +
      Math.floor(date.getTime() + date.getSeconds() / 2),
    description: 'Risk Report Download',
  },
 };
```

Figure 190: PDF reader 2

```
const historyDownload=()=> {
  //Function to check the platform
  //If iOS the start downloading
  //If Android then ask for runtime permission
  if (Platform.OS === 'ios') {
   DownloadHistory();
  } else {
    try {
      PermissionsAndroid.request(
        PermissionsAndroid.PERMISSIONS.WRITE_EXTERNAL_STORAGE
      ).then(granted => {
        if (granted === PermissionsAndroid.RESULTS.GRANTED) {
          //Once user grant the permission start downloading
          console.log('Storage Permission Granted.');
          DownloadHistory();
        } else {
          //If permission denied then show alert 'Storage Permission

         Alert.alert('storage_permission');
        }
      });
    } catch (err) {
      //To handle permission related issue
      console.log('error', err);
    }
  }
}
```

Figure 189: PDF reader 3

```
return (
  <Screen style={ROOT} preset="scroll">
    <View style={TOOLS_PDF}>
    <Icon
        name='long-arrow-left'
        type='FontAwesome'
        size={25}
        color='black'

        style={{ marginHorizontal: metrics.widthPercentageToDP(10),
          margin : metrics.heightPercentageToDP(2)}}
          onPress={()=>navigation.goBack()}
          />
    <Icon
        name='share'
       // type='FontAwesome'
        size={40}
        color='orange'

        style={ICON_STYLE}
        onPress={async () => { await share() } }   />
   <Icon
        name='search'
       // type='FontAwesome'
        size={40}
        color='orange'
        style={ICON_STYLE}

          />
           <Icon
        name='download'
```

Figure 192: PDF reader 4

```
    </View>
    <View>
    <Pdf
                  source={source}
                  onLoadComplete={(numberOfPages) => {
                      console.log(`Number of pages: ${numberOfPages}`);
                  }}
                  onPageChanged={(page) => {
                      console.log(`Current page: ${page}`);
                  }}
                  onError={(error) => {
                      console.log(error);
                  }}
                  onPressLink={(uri) => {
                      console.log(`Link pressed: ${uri}`);
                  }}
                  style={PDF_View}/>
                  </View>

  </Screen>
 )
})
```

Figure 191: PDF reader 5

Screens :

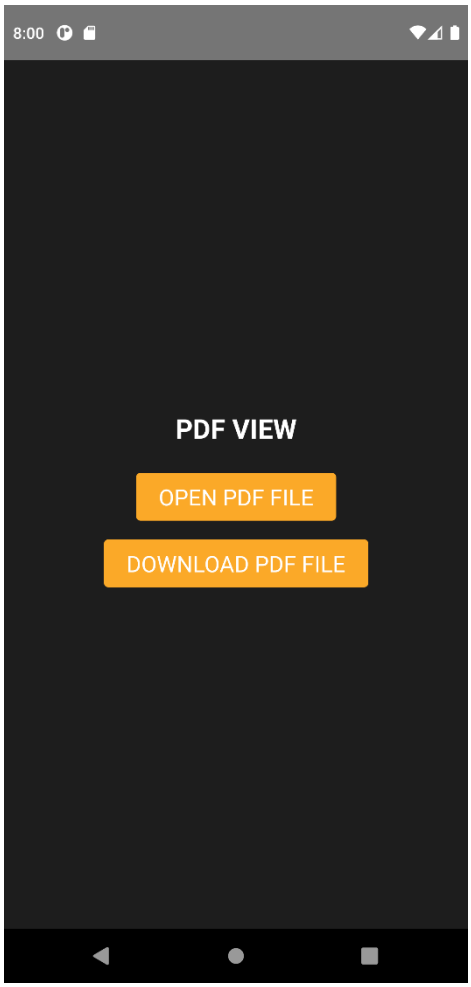


Figure 193: PDF view screen

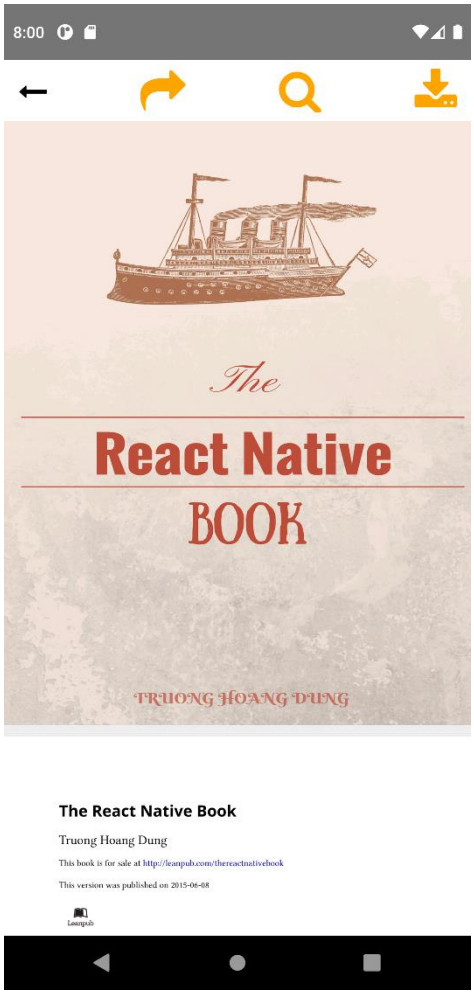


Figure 194: PDF opened

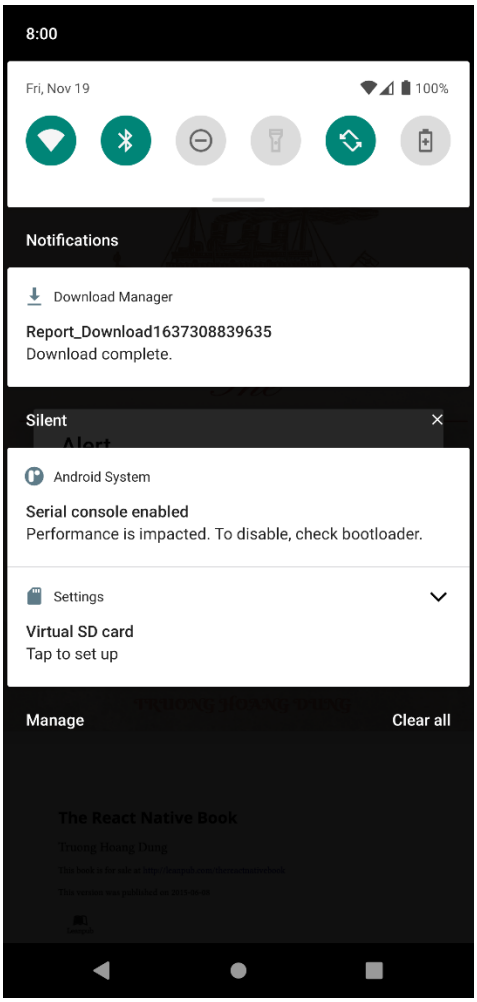


Figure 196: Download PDF 1

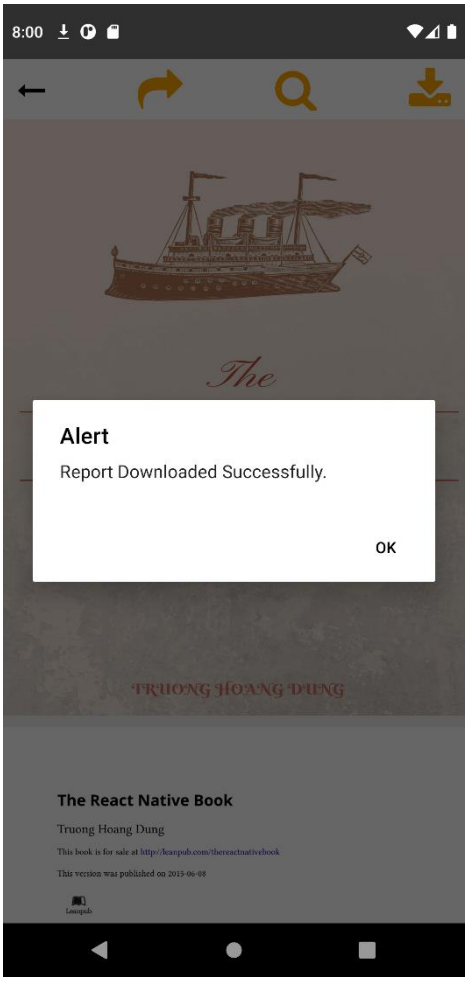


Figure 195 :download PDF 2

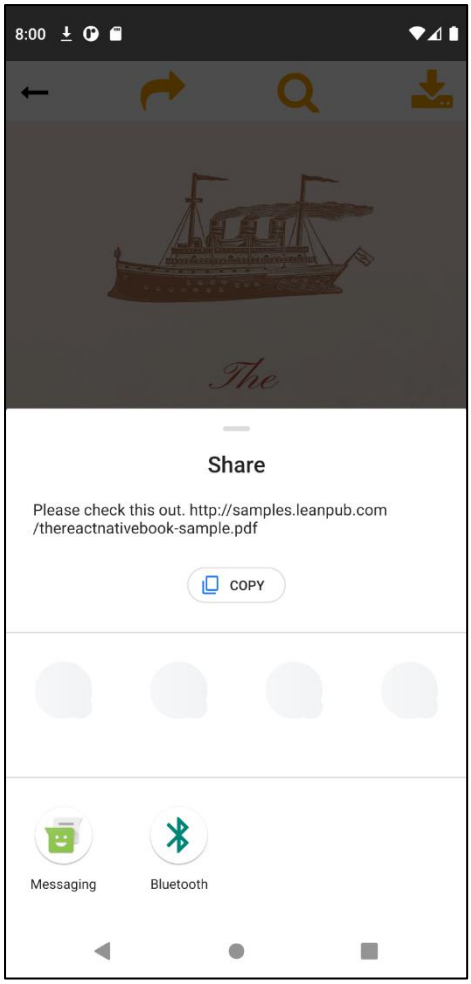


Figure 197: share PDF

Chapitre 32: Géolocalisation Avec Google Maps

Introduction :

react-native-maps est un pakcage qui aide a géolocaliser en real-time les appareils avec GOOGLE MAPS.

L'architecture de projet :

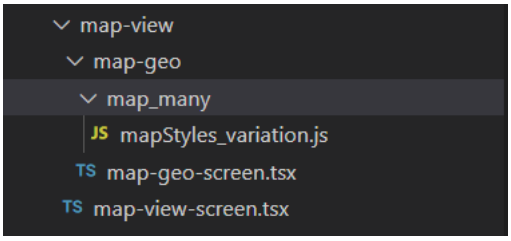


Figure 198: map arhcitecture

```
import { color } from "../../theme"
import metrics from "../../theme/metrics"
export const MapViewScreen = observer(function MapViewScreen() {
  const navigation = useNavigation()
  return (
    <Screen style={ROOT} preset="scroll">
    <Text preset="header" text="MAP VIEW" style={{ alignSelf:'center',margin: metrics.heightPercentageToDP(2)}} />
    <Button text="SEE MAP GEOLOCALISATION"  style={Button_Style} textStyle={Button_Text}
    onPress={()=>navigation.navigate("map_geo")}
    />
    </Screen>
  )
})
const ROOT: ViewStyle = {
  backgroundColor: color.palette.black,
  flex: 1,
  alignItems:'center',
  justifyContent:'center',
}
const Button_Style : ViewStyle={
  alignSelf:'center',
  margin: metrics.widthPercentageToDP(2)
 }
 const Button_Text:TextStyle={
   fontSize:20,
 }
```

Figure 199: map geo view

```
const [mapStyles,setMapStyles]=useState(Standard)
  const Map_View =()=>{
return(

    <MapView

provider={PROVIDER_GOOGLE}
style={styles.map}
customMapStyle={mapStyles}
        showsUserLocation={true} >
<Marker
            draggable
            coordinate={{
              latitude: 33.586050381932075,
              longitude:  -7.62976720854807,

            }}
           }
            onDragEnd={
              (e) => alert(JSON.stringify(e.nativeEvent.coordinate))
            }

            title={'Gear9 CASABLANCA'}
            description={'219 Bd Zerktouni, angle Bd Brahim Roudani, Casablanca 20000'}
>
      <View>
    <Image source={{uri : "https://cdn-icons.flaticon.com/png/512/2377/premium/2377922.png?token=exp=1637334936~hmac=fb55cd244c2c62c66c10199bb1b18a92"}}
      style={PIN_MARKER_STYLE}
    />
    </View>
  </Marker>
  <Marker
```

Figure 200: map geo screen 1

```
<Marker
            draggable
            coordinate={{
              latitude: 34.022357021074,
              longitude:  -5.0075797678306015,

            }}
          }
            onDragEnd={
              (e) => alert(JSON.stringify(e.nativeEvent.coordinate))
            }

            title={'Gear9 FES'}
            description={'Angle Rue Tarik Ibn Ziad Et Rue Abdelkrim Benjelloun 6ème Etage, Commune Agdal, Fès'}

    <View>
  <Image source={{uri : "https://cdn-icons.flaticon.com/png/512/2377/premium/2377922.png?token=exp=1637334936~hmac=fb55cd244c2c62c66c10199bb1b18a92"}}
    style={PIN_MARKER_STYLE}
  />
  </View>
</Marker>



/MapView>

;
    }
const Polygone_View =()=>{
  return (
```

Figure 201: map geo screen 2

```
const MAPS=()=>{
  return (
    <Callout style={MAPS_STYLE}>

    <View style={ROW_STYLES_MAP}>

    <TouchableOpacity style={MAP_STYLE_PAGE} onPress={()=>setMapStyles(Night)}>
    <Icon  name ="moon"  color="black" size={30}  />
    <Text style={MAP_STYLE_TEXT}>Night</Text>
    </TouchableOpacity>
    <TouchableOpacity style={MAP_STYLE_PAGE} onPress={()=>setMapStyles(Retro)}>
    <Icon name ="adjust"  color="black" size={30} />
    <Text style={MAP_STYLE_TEXT}>Retro</Text>

    </TouchableOpacity>
    <TouchableOpacity style={MAP_STYLE_PAGE} onPress={()=>setMapStyles(Standard)}>
    <Icon name ="sun"  color="orange" size={30} />
    <Text style={MAP_STYLE_TEXT}>Standard</Text>

    </TouchableOpacity>



    </View>

            </Callout>
  );
}
```

Figure 202: map geo screen 3

Pour changer le style de Map on utilise le site web https://mapstyle.withgoogle.com/

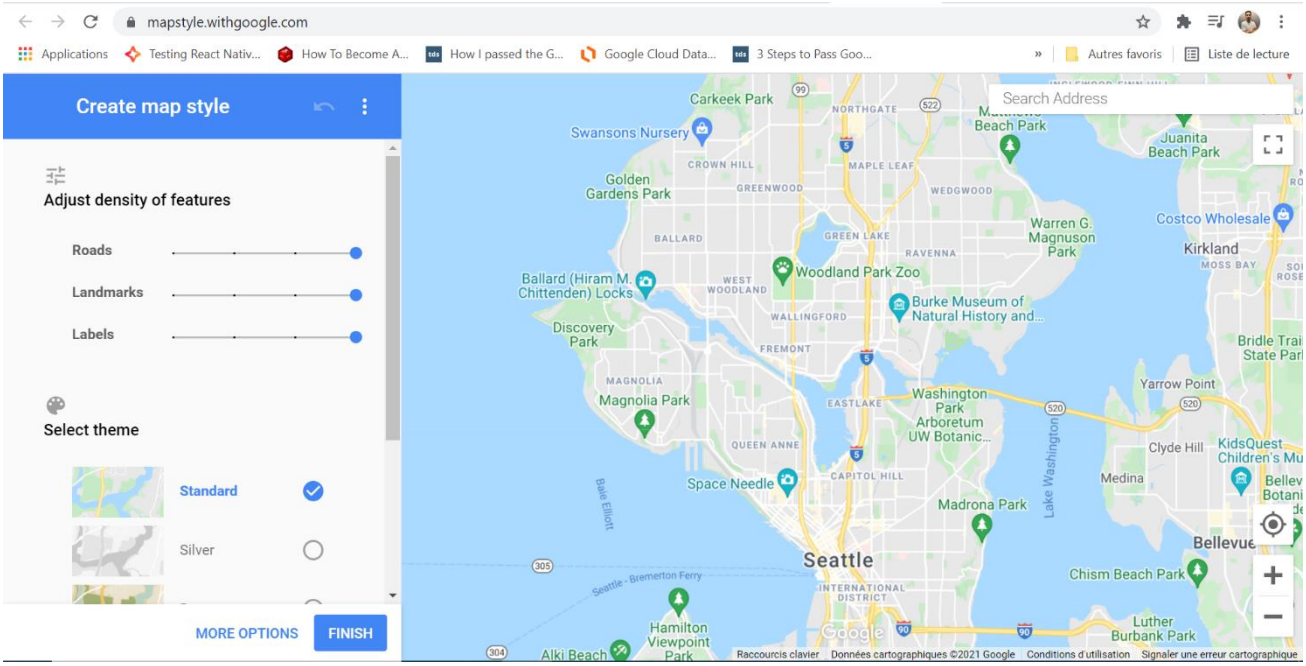


Figure 203 : map customized 1

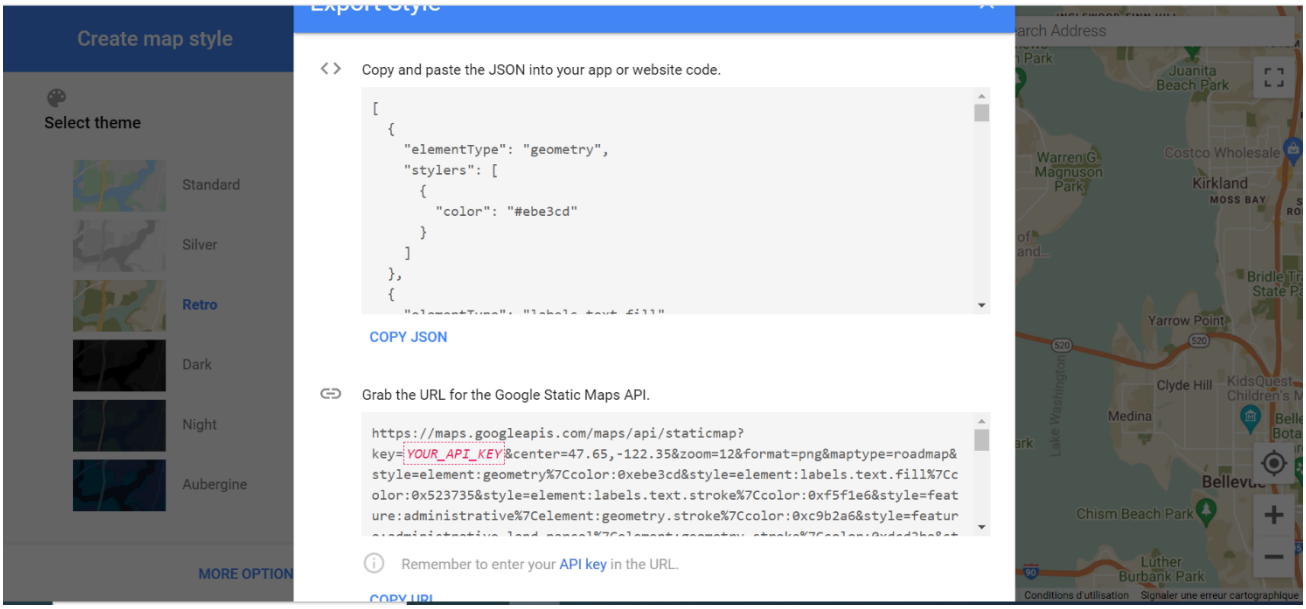


Figure 204: map customized 2

Finalement sur le site on copie le fichier json et on l'importer sur le projet .

```
const Standard = [
  {
    "featureType": "administrative.country",
    "elementType": "geometry",
    "stylers": [
      {
        "color": "#3dbeff"
      },
      {
        "weight": 2
      }
    ]
  }
]
const  Night=[
  {
    "elementType": "geometry",
    "stylers": [
      {
        "color": "#242f3e"
      }
    ]
  },
  {
    "elementType": "labels.text.fill",
    "stylers": [
      {
        "color": "#746855"
      }
    ]
  },
  {
```

Screens :

Figure 206: map marker gear9 CASABLANCA

Figure 205:map marker gear9 FES

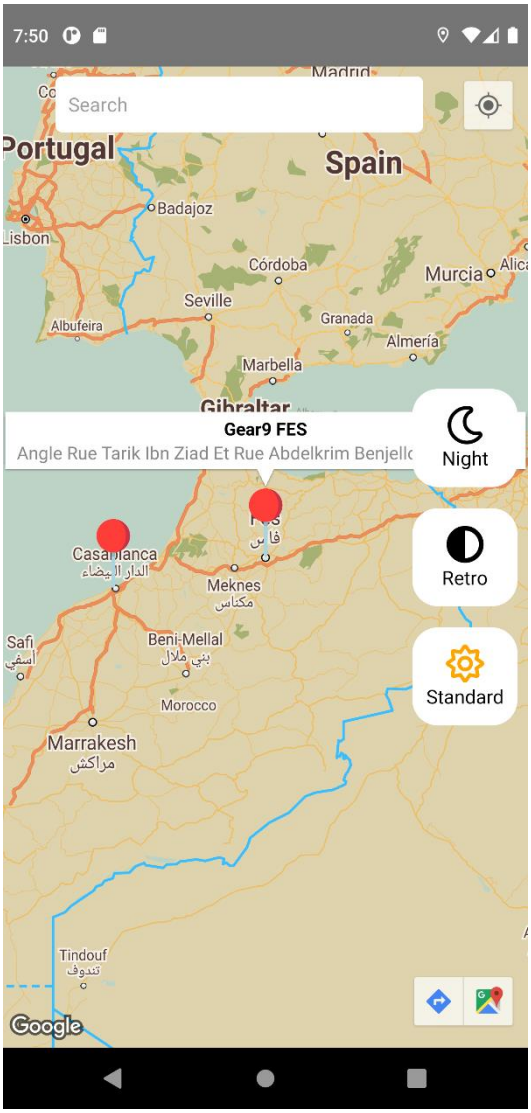


Figure 208:map style retro

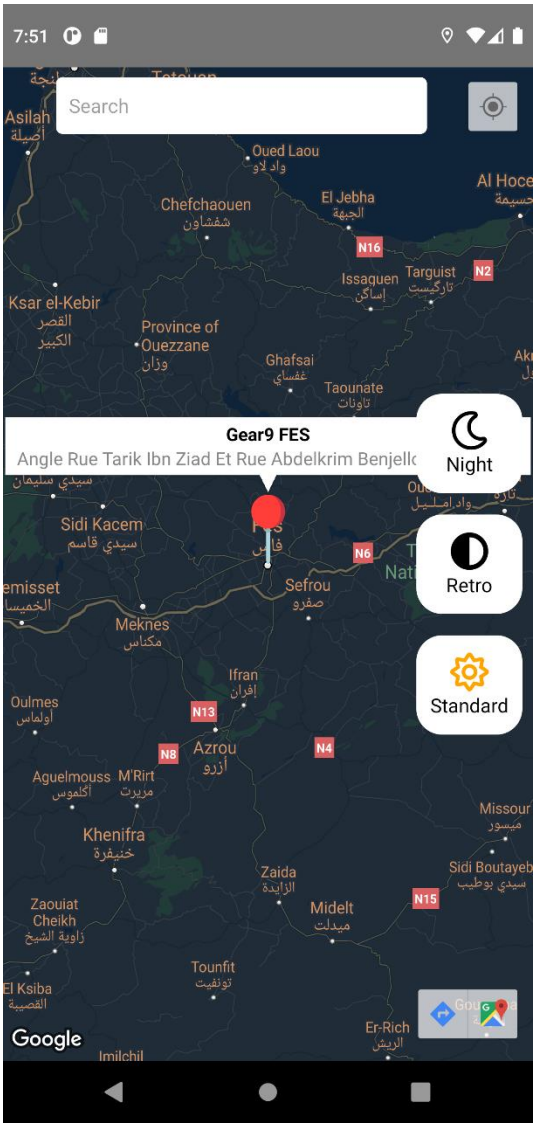


Figure 207: map style night

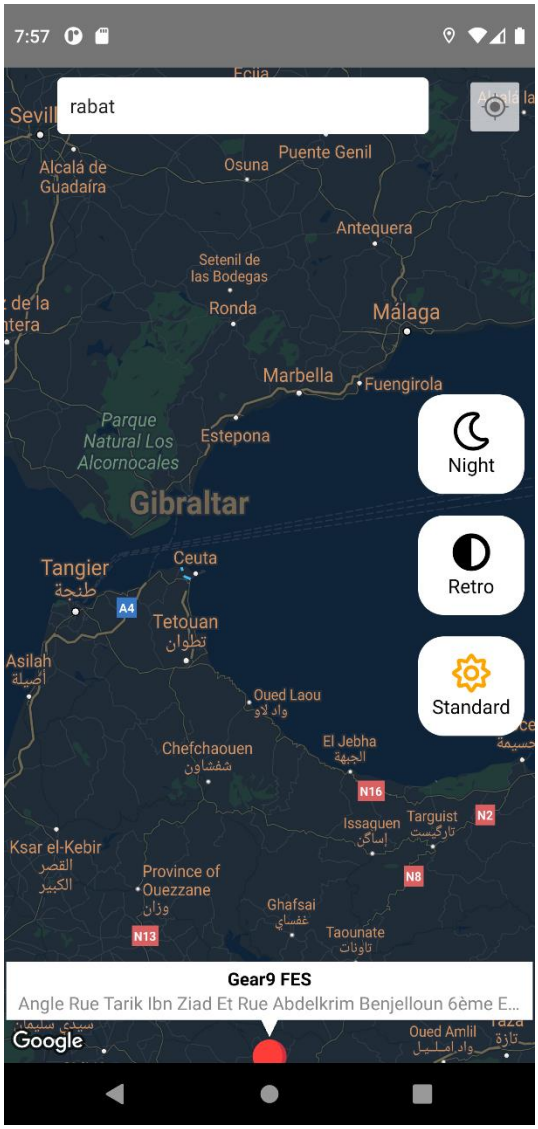


Figure 209: map autocomplete search

Chapitre 33: QR Code avec React-Native Download /Scan/Convert

Introduction :

QR Code svg est basé sur react native svg .

Architecture du projet :

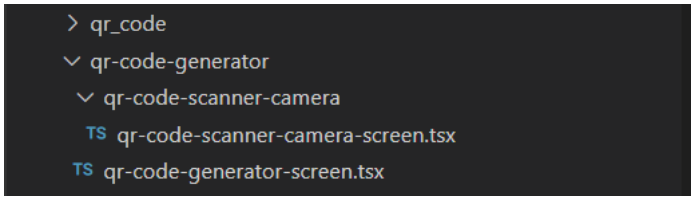


Figure 210: qr code architecture

Qr Code Generator :

```
export const QrCodeGeneratorScreen = observer(function QrCodeGeneratorScreen() {
  // Pull in one of our MST stores
  // const { someStore, anotherStore } = useStores()

  // Pull in navigation via hook
  const navigation = useNavigation()
  const [qrValue,SetQRValue]=useState("hello");
  const Qr_Value_Validate=(newValue)=>{
      if (newValue===''){
        Alert.alert("qr code value shouldnt be empty enter a not emply value !")
      }
      else{
        SetQRValue(newValue);

      }
  }
  const [svg,setSvg]=useState("");

  useEffect(()=>{
    //console.log(svg1)
  });
  return (
    <Screen style={ROOT} preset="scroll">

      <Text preset="header" text="Qr Code"  style={{alignSelf:'center'}}/>
      <View style={INPUT_QR}>
      <TextInput
      placeholder='put url to convert to qr code '
      style={{fontSize:15}}
      onChangeText={(value)=>Qr_Value_Validate(value)}
      />
      </View>
```

Figure 211: qr code generator 1

```
        />
        </View>

        <View style={QR_STYLE}>
        <QRCode

        value={qrValue}
        size={300}
        //getRef={c => (svg1 = c)}
      />
        </View>
        <View style={QR_TOOLS_ROW}>
        <TouchableOpacity style={SCANNER_QR} onPress={()=>navigation.navigate("qr_code_scanner")}>
          <Text>Open Scanner</Text>
        <Icon  name ="camera"  color="white" size={40}  />

        </TouchableOpacity>
        <TouchableOpacity style={SCANNER_QR} onPress={()=>{console.log("hello")}}>
          <Text>Share Qr code</Text>
        <Icon  name ="share"  color="orange" size={40}  />

        </TouchableOpacity>
        </View>
      </Screen>
    )
  })
```

Figure 213: qr code generator 2

Qr Code Scanner :

```
  let svg1= useRef("");
const [imageSave,SetSave]=useState({ busy: false, imageSaved: true  })
const saveQrToDisk=()=> {
  svg1.toDataURL((data) => {
    RNFS.writeFile(RNFS.CachesDirectoryPath+"/some-name.png", data, 'base64')
      .then((success) => {
        return CameraRoll.saveToCameraRoll(RNFS.CachesDirectoryPath+"/some-name.png", 'photo')
      })
      .then(() => {
        SetSave({ busy: false, imageSaved: true  })
        ToastAndroid.show('Saved to gallery !!', ToastAndroid.SHORT)
      })
  })
}
  const Read_Qr_Code= e =>{
    ///console.log(e.data);
    Linking.openURL(e.data);
      setScanned(e.data)
  }
  const OpenQrCodeUrl=(url)=>{
    Linking.openURL(url)
  }
  const [scanned,setScanned]=useState("https://www.youtube.com/watch?v=iJzJ7d5CK4A");
  return (
    <Screen style={ROOT} preset="scroll">
      <Text preset="header" text="QR Code Scanner" style={{alignSelf:'center'}} />
      <QRCodeScanner
      onRead={Read_Qr_Code}
      showMarker={true}
      customMarker={
        <View >
      <Icon  name ="scan-outline"  color="orange" size={300}   />
```

Figure 212: qr code scanner 1

```
const [scanned,setScanned]=useState("https://www.youtube.com/watch?v=iJzJ7d5CK4A");
return (
  <Screen style={ROOT} preset="scroll">
    <Text preset="header" text="QR Code Scanner" style={{alignSelf:'center'}} />
    <QRCodeScanner
    onRead={Read_Qr_Code}
    showMarker={true}
    customMarker={
      <View >
    <Icon  name ="scan-outline"  color="orange" size={300}   />

      </View>
    }
    />
    <View style={URL_SCANNED}>
      <Text>URL Scanned : {scanned}</Text>
    <Button text="open url" style={{alignSelf:'center'}} textStyle={{fontSize:20}} onPress={()=>OpenQrCodeUrl(scanned)} />
    </View>


  </Screen>
)
})
```

Figure 214: qr code scanner 2

Screens :

Figure 216: qr code is not empty

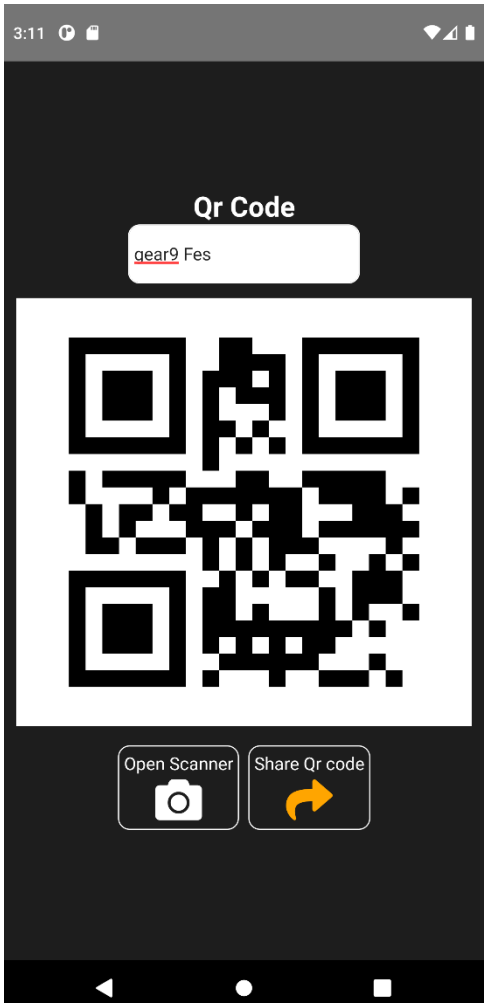


Figure 215:qr code converter

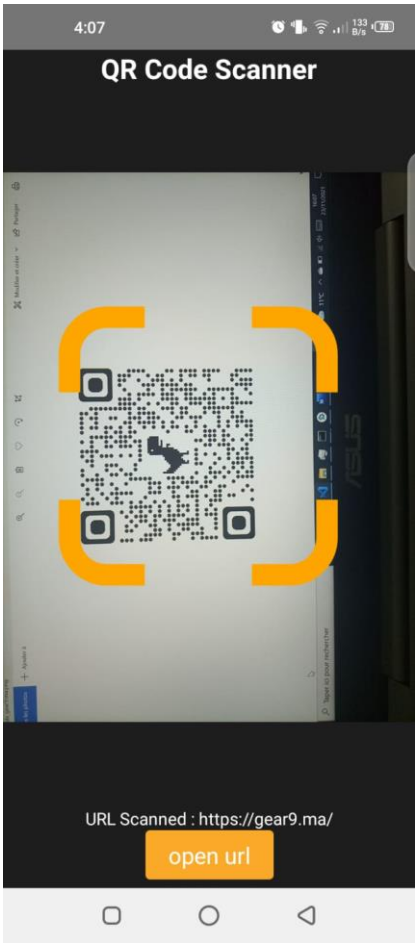


Figure 218: Scan Qr Code

Figure 217: Open Url scanned

Save Qr Code dans la gallérie :

```
let svg=useRef("");
const SaveQrCode=()=>{
  svg.toDataURL((data)=>{

    RNFS.writeFile(RNFS.CachesDirectoryPath+`/qr.png`,data,"base64" )
    .then((success)=>{

      return CameraRoll.save(RNFS.CachesDirectoryPath+`/qr.png`,"photo" );

    })
    .catch((e)=>{
      console.log("saveToGallery",e)
    })
  })
}
```

Figure 219: save qr code

Screens :

Figure 221: save qr code to gallery

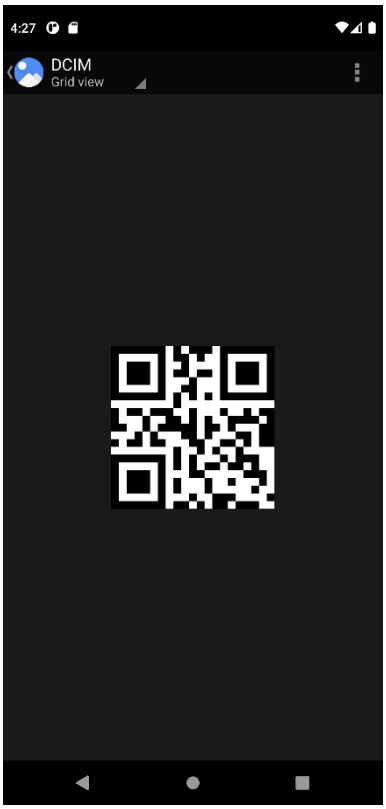


Figure 220: display qr code in gallery