**Introduction à React native**

**Qu'est-ce que React Native ?**

Développé par la société Meta (anciennement Facebook) en 2015, React Native est un framework JavaScript open-source, hybride et multi-plateforme, dérivé de ReactJS. Ce framework permet de créer des applications mobiles véritablement natives pour iOS et Android à partir d’une seule base de code, sans compromettre l’expérience utilisateur.

Le fonctionnement repose sur un pont (Bridge) qui permet au code React (écrit en JavaScript) d’interagir dynamiquement avec les API natives du système d’exploitation. Les composants graphiques générés sont donc entièrement natifs, garantissant des performances et une expérience utilisateur comparables à celles des applications développées directement en Swift ou Kotlin.

**Applications hybrides et le rôle de React Native**

Une application hybride se positionne entre une application web et une application native :

* Elle est **moins coûteuse et plus rapide** à développer qu’une application native, car elle utilise une seule base de code pour plusieurs plateformes.
* Elle offre **plus de flexibilité et de fonctionnalités** qu’une application web, notamment grâce à son accès aux fonctionnalités matérielles des appareils (GPS, caméra, etc.).
* La **maintenance** est simplifiée, car il n’y a qu’un seul code à maintenir, contrairement à des applications natives séparées pour iOS et Android.

**Adoption et succès de React Native**

Depuis sa création, React Native a été adopté par de nombreuses grandes entreprises grâce à ses avantages en termes de développement rapide et d’efficacité. Voici quelques exemples :

* Meta utilise React Native pour ses propres applications telles que Facebook, Instagram, et Messenger.
* Airbnb a intégré cette technologie pour ses applications de réservation.
* Microsoft utilise React Native dans des projets tels que Skype et Teams.
* Discord, Coinbase, Uber Eats, et bien d’autres entreprises y ont recours pour leurs besoins mobiles.

En raison de son évolutivité, de son écosystème riche et de sa capacité à combiner les avantages des applications natives et hybrides, React Native est aujourd’hui l’un des frameworks les plus populaires pour le développement mobile.

**Différences entre React et React Native**

Bien que React et React Native partagent des bases communes, notamment leur architecture et leur utilisation de JavaScript, ils ont des différences fondamentales liées à leurs objectifs et contextes d’utilisation.

1. Objectif

* React :  
  React (ou ReactJS) est une bibliothèque JavaScript utilisée principalement pour créer des interfaces utilisateur (UI) sur le web. Elle se concentre sur le rendu des éléments HTML dans un navigateur, en offrant une expérience utilisateur fluide grâce au DOM virtuel.
  + Exemple d’utilisation : Sites web interactifs comme Facebook ou Netflix.
* React Native :  
  React Native est un framework basé sur React, conçu spécifiquement pour le développement d’applications mobiles natives. Contrairement à React, il ne manipule pas le DOM, mais utilise des composants graphiques natifs pour chaque plateforme (iOS et Android).
  + Exemple d’utilisation : Applications mobiles comme Instagram ou UberEats.

2. Plateformes cibles

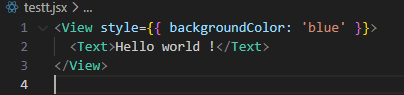
* React : Exclusivement pour le web, exécuté dans des navigateurs (Chrome, Firefox, Safari, etc.).
* React Native : Dédié aux applications mobiles, compilées pour fonctionner sur des systèmes natifs tels qu’iOS et Android.

3. Langage de rendu

* React : Utilise le HTML (et JSX) pour décrire la structure et le rendu des composants. Ces derniers sont traduits en balises HTML et styles CSS dans le navigateur.
  + Exemple :



* React Native : Utilise des composants spécifiques aux plateformes au lieu d’éléments HTML. Ces composants (comme <View>, <Text>, ou <Button>) sont traduits en éléments natifs par le framework.
  + Exemple :



**Installation et Setup de React Native avec Android Studio**

**1. Prérequis**

Avant de commencer, assurez-vous que vous avez installé les éléments suivants sur votre machine :

* Node.js : React Native nécessite Node.js pour fonctionner. Vous pouvez télécharger et installer Node.js [ici](https://nodejs.org/).
* npm : npm (Node Package Manager) est installé avec Node.js. Vérifiez son installation en exécutant la commande suivante dans le terminal :

*npm -v*

* Java Development Kit (JDK) : Android Studio nécessite le JDK. Téléchargez-le et installez-le depuis [Oracle](https://www.oracle.com/java/technologies/javase-jdk11-downloads.html) si ce n'est pas déjà fait.
* Android Studio : Téléchargez et installez Android Studio depuis [ici](https://developer.android.com/studio). Assurez-vous d'inclure le SDK Android et l'AVD Manager pendant l'installation.

**2. Installation de React Native CLI**

1. Installer React Native CLI en exécutant la commande suivante dans votre terminal :

*npm install -g @react-native-community/cli*

Cela installe la CLI de React Native globalement sur votre machine.

**3. Initialisation d'un Nouveau Projet React Native**

1. Créer un nouveau projet React Native avec la commande npx :

*npx @react-native-community/cli init monProjet*

Cela va initialiser un nouveau projet React Native nommé projetTech et créer un dossier contenant la structure du projet.

**4. Installation des Dépendances**

1. Une fois que le projet est créé, allez dans le répertoire du projet :

*cd monProjet*

1. Installer les dépendances nécessaires en exécutant la commande suivante :

*npm install*

5. **Configuration de l'Emulateur Android**

1. Ouvrez Android Studio et configurez un émulateur Android via le AVD Manager.
2. Sélectionnez un appareil virtuel (par exemple, Pixel 4) et lancez l'émulateur.

**6. Démarrer l'Application sur l'Émulateur**

1. Lancer l'application sur l'émulateur en exécutant la commande suivante dans le répertoire du projet :

*npx react-native run-android*

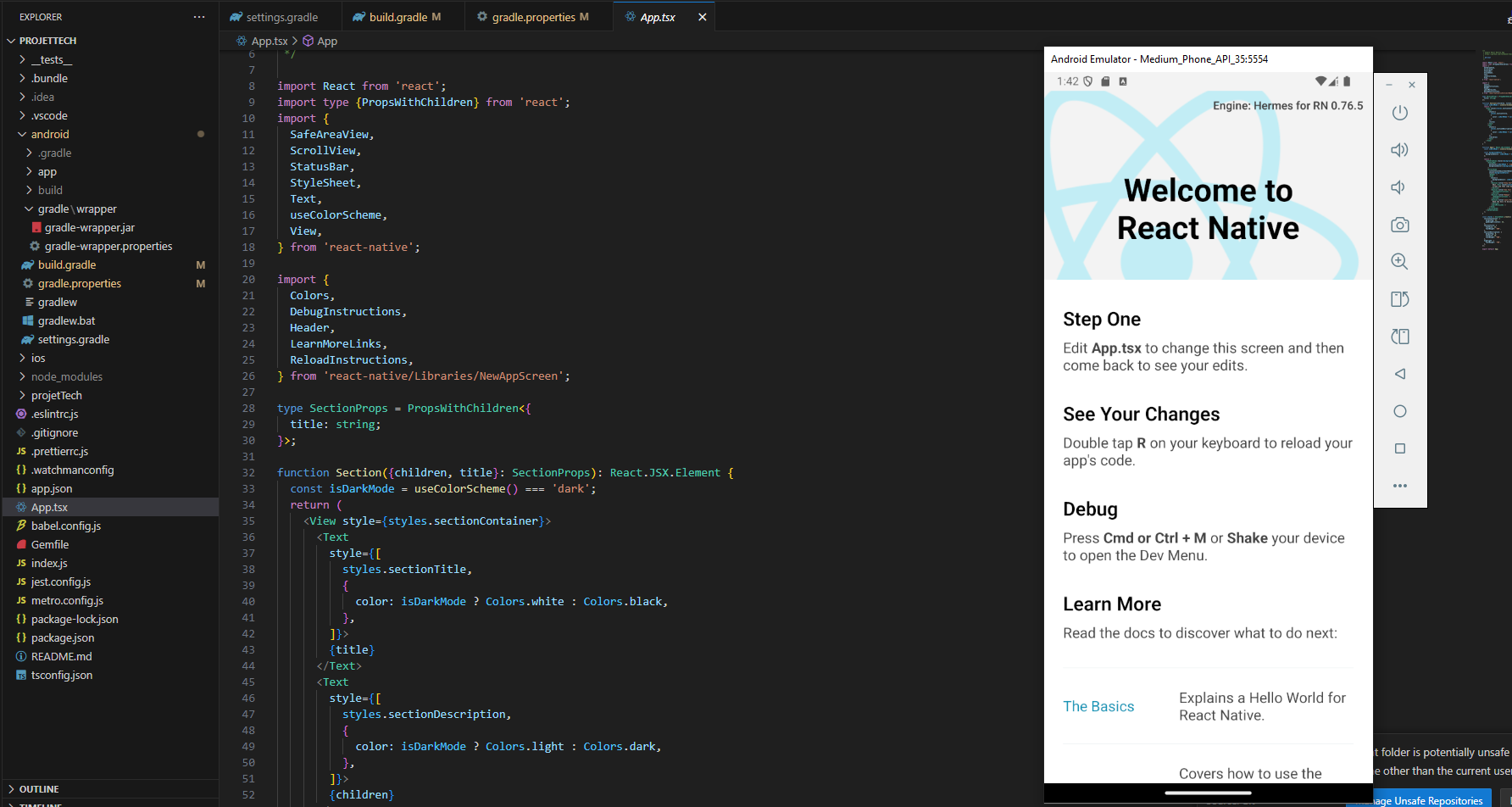
Cette commande compile l'application et la déploie sur l'émulateur Android que vous avez lancé.

**7. Tester et Développer**

* Si l'application se lance correctement sur l'émulateur, vous pouvez commencer à la modifier dans VS Code ou votre éditeur de code préféré.
* Les modifications seront automatiquement rechargées dans l'émulateur si le rechargement automatique est activé. Vous pouvez appuyer sur Ctrl + R dans l'émulateur pour forcer un rechargement manuel.

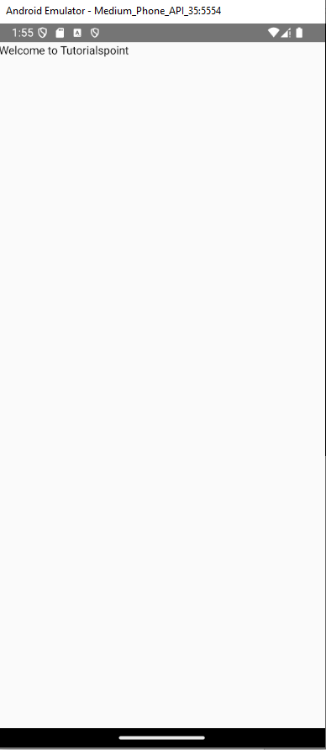
8**. Résolution de Problèmes Communs**

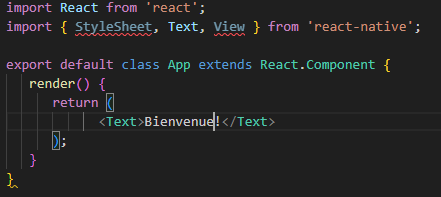
* Si le projet ne se lance pas sur l'émulateur : Assurez-vous que l'émulateur est en cours d'exécution avant de lancer la commande npx react-native run-android.
* Si des erreurs de dépendances surviennent : Essayez de réinstaller les modules en exécutant npm install ou npm audit fix pour corriger les problèmes de vulnérabilité.

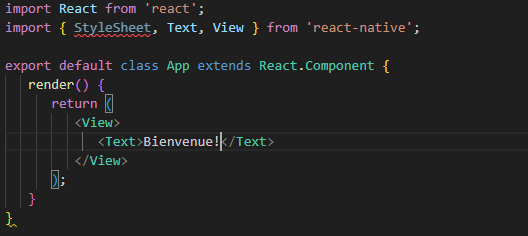


**Apprenons le langage!**

**1. Commençons tout d’abord par comprendre les balises de base de react Native.**

Pour insérer un texte à afficher sur l’émulateur nous utiliserons la balise <Text></Text>.

Ex :

Il est préférable d’utiliser une balise <View></View> afin de regrouper d’autres composants pour mieux structuer l’interface. <View> est l’équivalent de d’un <div> en html. De plus, on peut appliquer des styles à cette balise en utilisant un objet StyleSheet (à voir après). 

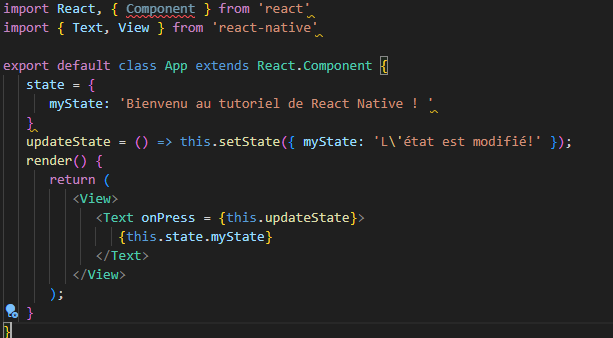
1. **State et Props**

Les données dans les composants React sont gérées par state et props. Ces deux concepts sont fondamentaux pour la gestion des informations dans une application React.

1. Différence entre State et Props

* State (état)  
  Le state est mutable, c’est-à-dire qu’il peut être modifié après sa création.  
  Il appartient au composant qui le déclare et est utilisé pour stocker et gérer les données locales propres à ce composant.
* Props (propriétés)  
  Les props sont immutables, ce qui signifie qu’elles ne peuvent pas être modifiées après leur transmission. Elles sont transmises par un composant parent à un composant enfant et servent à communiquer des informations entre les composants.

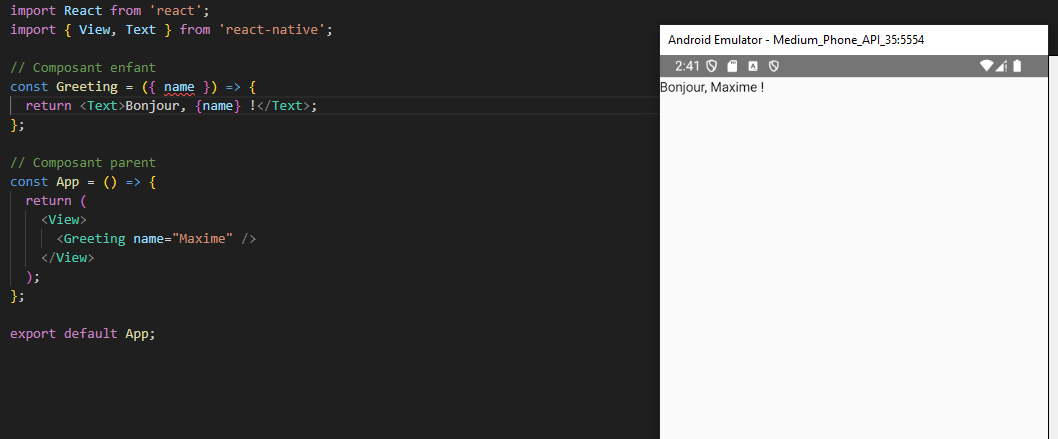
2. Utilisation de State

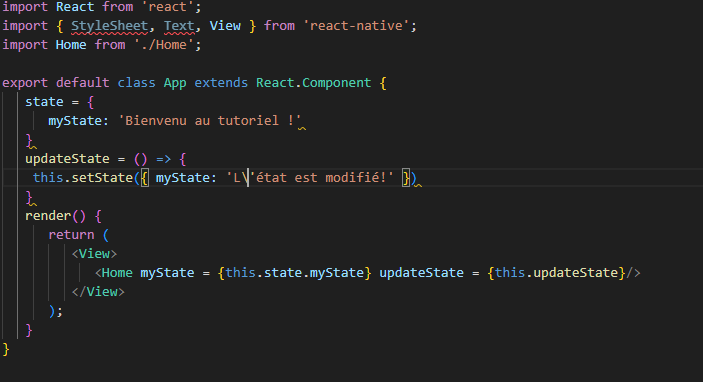
Le state est idéal pour des données qui changent avec le temps ou en réponse aux interactions de l’utilisateur. On utilise la fonction setState pour modifier la valeur d’une donnée

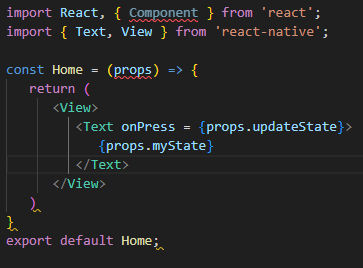
Résultat au clic :

1. Utilisation de Prop

Les props permettent de transmettre des données ou des fonctions d’un parent à un enfant. Un enfant ne peut pas modifier les props qu’il reçoit.

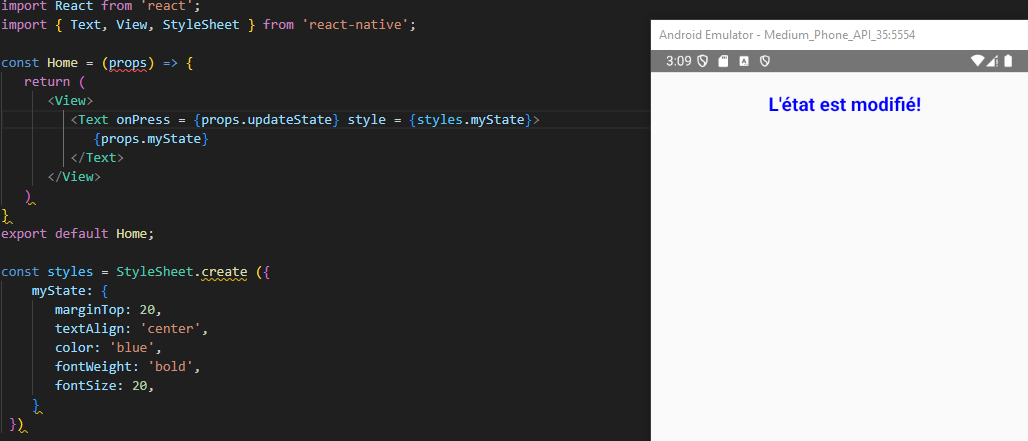


Exemple d’utilisation de state et prop :

 App.tsx Home.tsx

**3. Style**

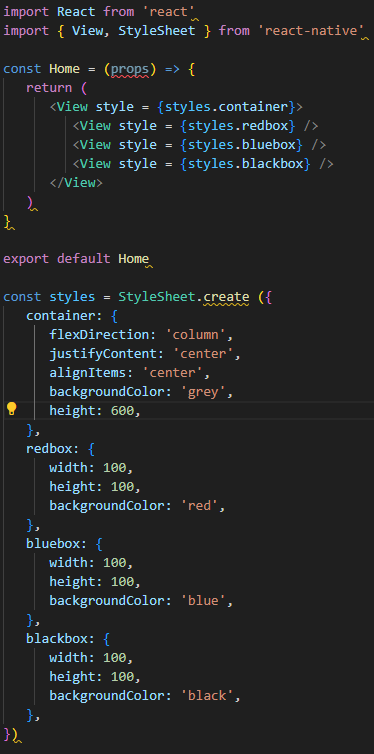
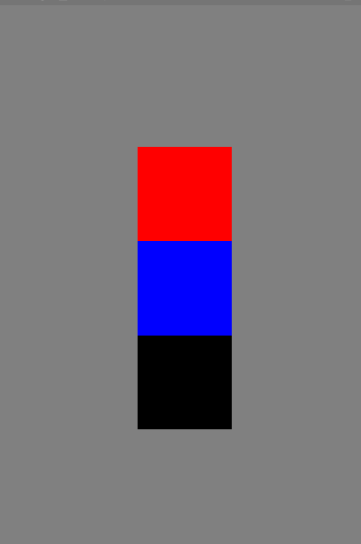
On peut donc stylisé notre application en utilisant l’objet StyleSheet. Dans le bas de la page, on va créer notre stylesheet et appliquer des styles. Pour appliquer le style à notre Text, nous devons ajouter **style = {styles.myText}**



**4. Flexbox**

Pour accommoder sur différentes tailles d’écran, nous utiliserons Flexbox. Nous l’utiliserons dans l’objet stylesheet et l’appliquerons sur la balise <View>.

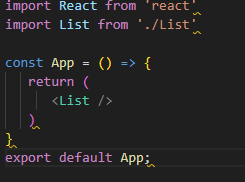
|  |  |  |
| --- | --- | --- |
| **Propriété** | **Valeurs** | **Description** |
| flexDirection | |  | | --- | |  |  |  | | --- | | column, row | | Spécifie si les éléments seront alignés verticalement ou horizontalement. |
| justifyContent | enter, flex-start, flex-end, space-around, space-between | Détermine comment les éléments doivent être distribués à l'intérieur du conteneur. |
| alignItems | center, flex-start, flex-end, stretched | Détermine comment les éléments doivent être alignés le long de l'axe secondaire (opposé à flexDirection). |

Exemple :

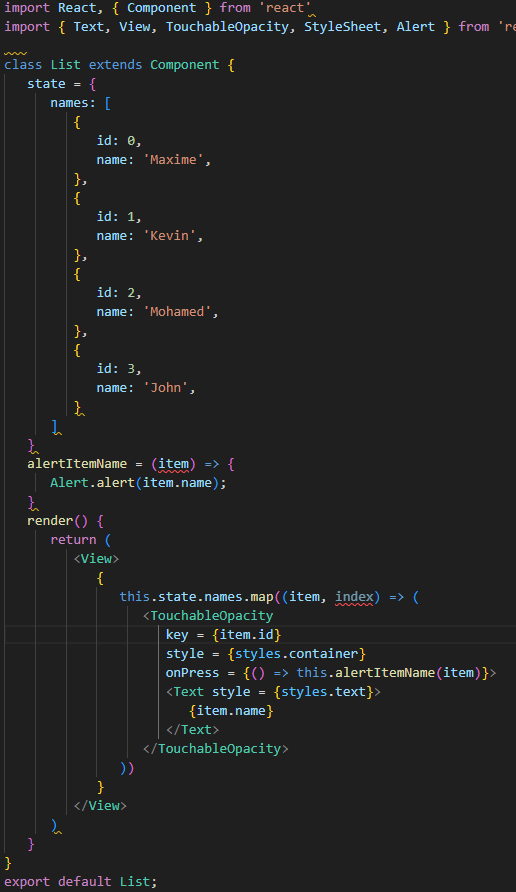
**5. ListView**

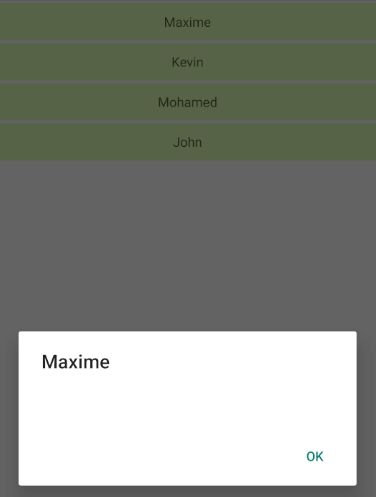
Dans cette section, nous allons voir comment créer et utiliser une list dans React Native.

App.tsx



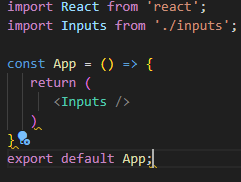
List.tsx



 Résultat :

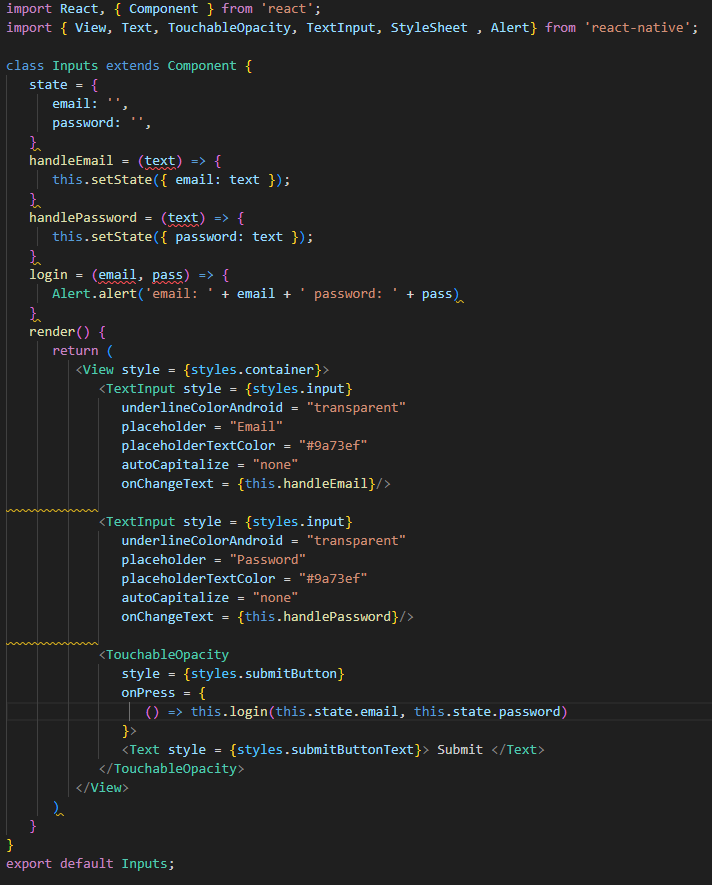
Dans cet exemple, nous avons utilisé deux nouveaux éléments (Alert et TouchableOpacity). Alert est un composant intégré qui permet d’afficher des alertes. TouchableOpacity permet de créer des zones cliquables et applique un effet visuel quand l’utilisateur clique dessus.

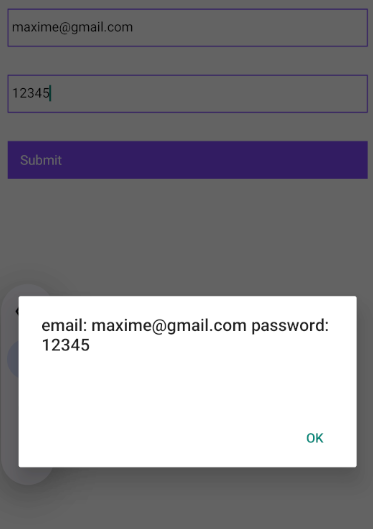
**6. Text Input**

Voyons maintenant comment utiliser un TextInput.

App.jsx

Inputs.tsx

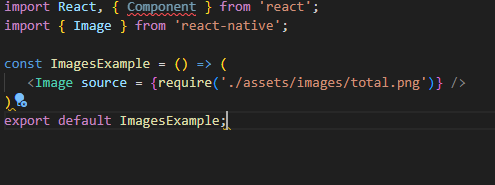




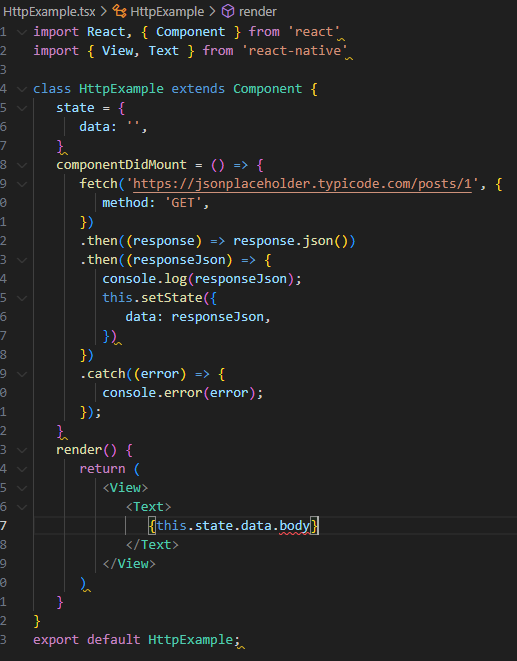
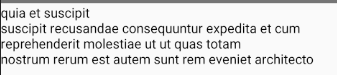
Quand on écrit dans les inputs, le state va être modifié et quand on clique sur Submit, le texte dans les inputs va être affiché dans le alert.

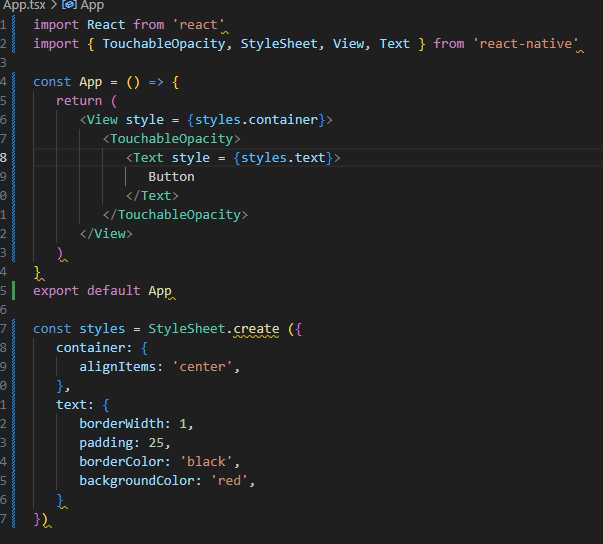
**6. Images**

Voyons maintenant comment utiliser une Image.

**7. HTTP**

Nous utiliserons fetch qui est une méthode pour envoyer des requêtes Http (GET, POST, etc) et récuperer des données. Nous verrons aussi componentDidMount qui est une méthode cycle. C'est une méthode du cycle de vie de React appelée une seule fois lorsque le composant est ajouté à l'écran. On l'utilise pour charger des données automatiquement dès que le composant apparaît.



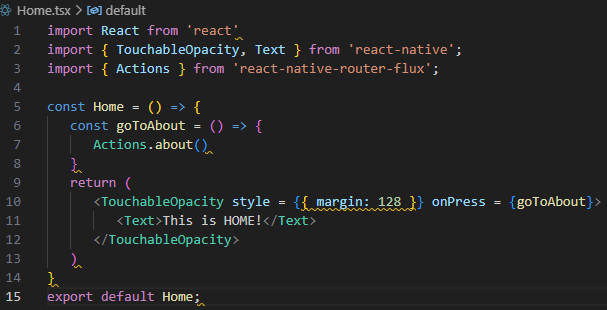
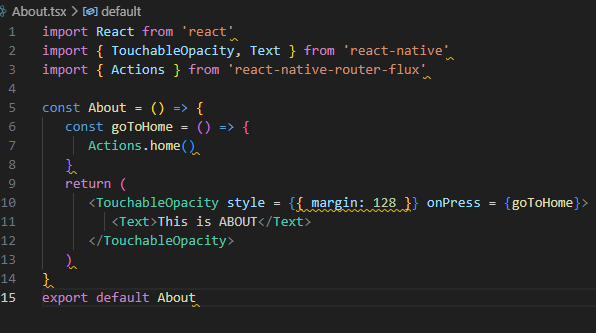
**8. Button**

**9. Router**

Pour la navigation entre les différentes pages nous utiliserons un router

App.tsx Routes.jsx

Home.tsx About.tsx



**Bonne chance !**