**REACT NATIVE**

Framework de JavaScript que permite crear aplicaciones nativas para IOS y Android. Esta basada en React, librarías de JavaScript creadas por Facebook que permite construir interfaces de usuario



**Librería:**

Colección de código ya escrito que permite ser rehusado para simplificar y acelerar el proceso de construcción de código. Permite de funciones o métodos que permiten hacer el trabajo mas rápido y eficiente.



**Framework:**

Herramientas o códigos pre escrito funciones , métodos y librerías que permite escribir aplicaciones mas rápido.

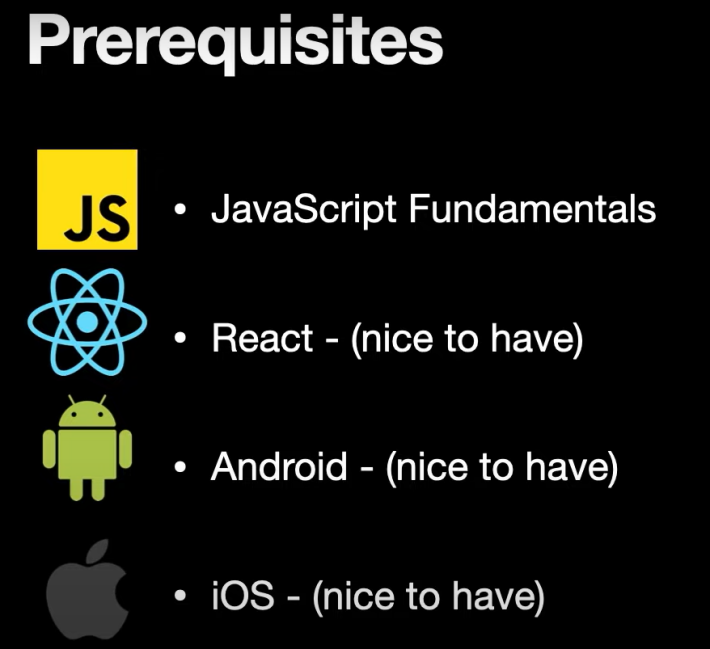




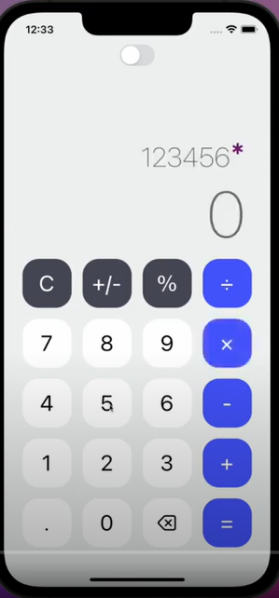
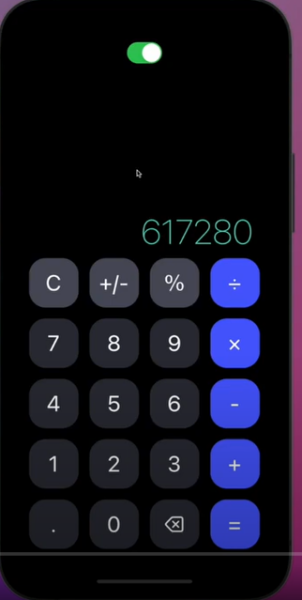


Ejemplo de limitaciones IOS puede tener algunas características que no se pueden con Android. (Aqui podrias requerir código nativo)

Podrias sentir que las APP nativas son un poco diferentes a las de React Native (por el punto anterior, por ejemplo)

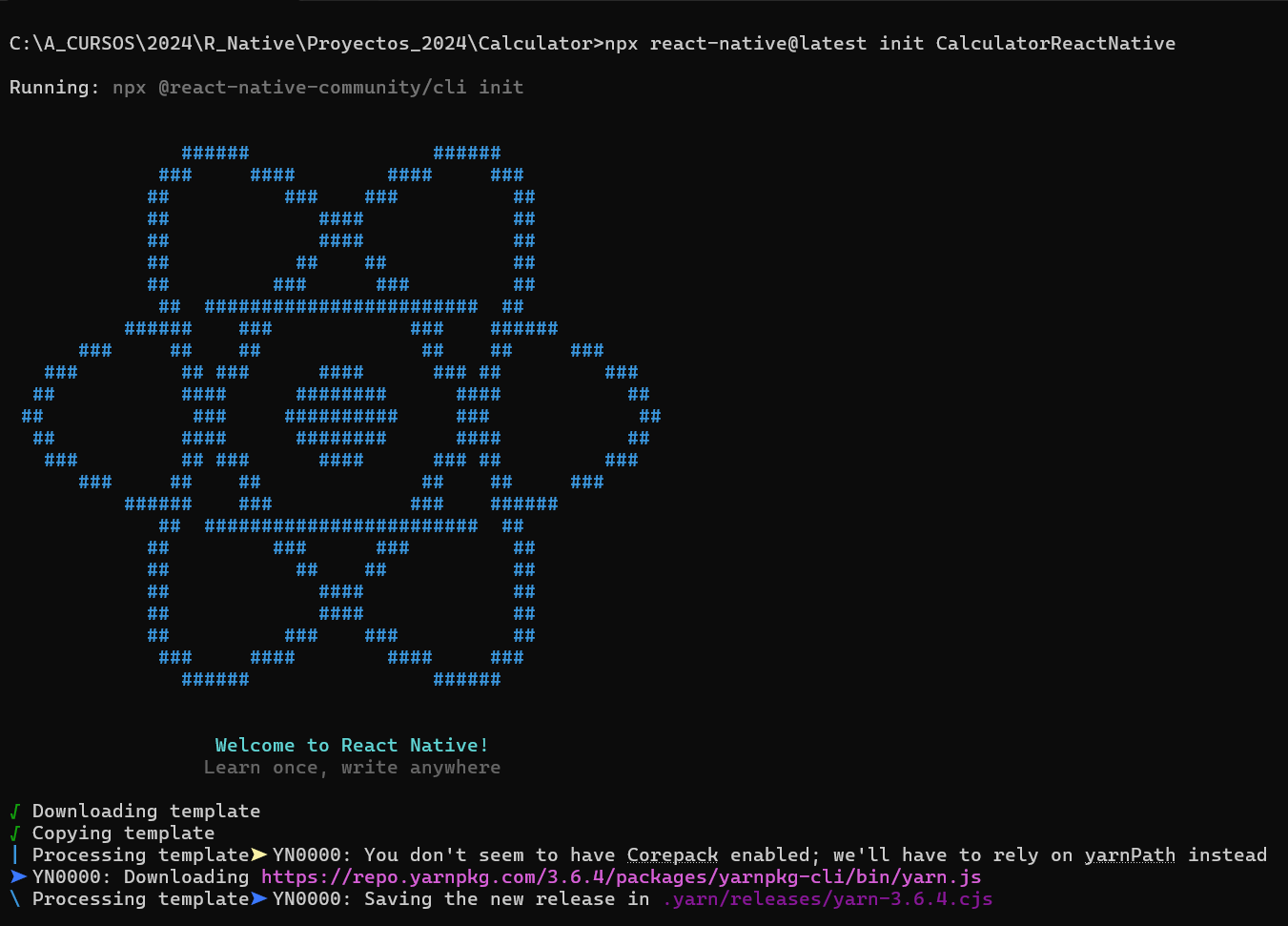


**Proyecto de Calculadora en React Native**

* Creamos un proyecto React Native con TypeScript básico

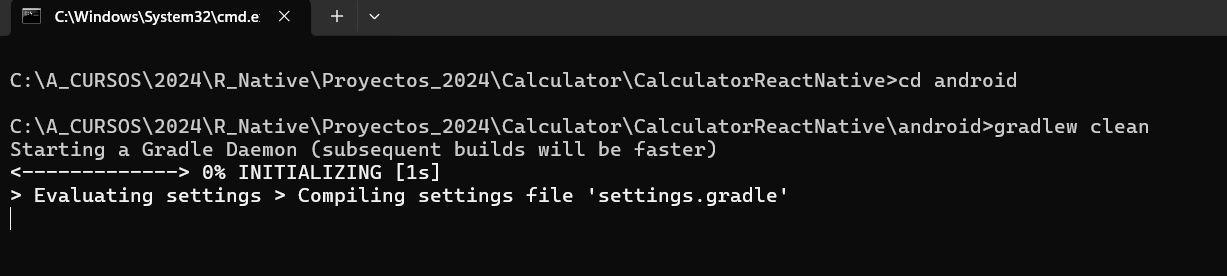
\_> npx react-native@latest init CalculatorReactNative



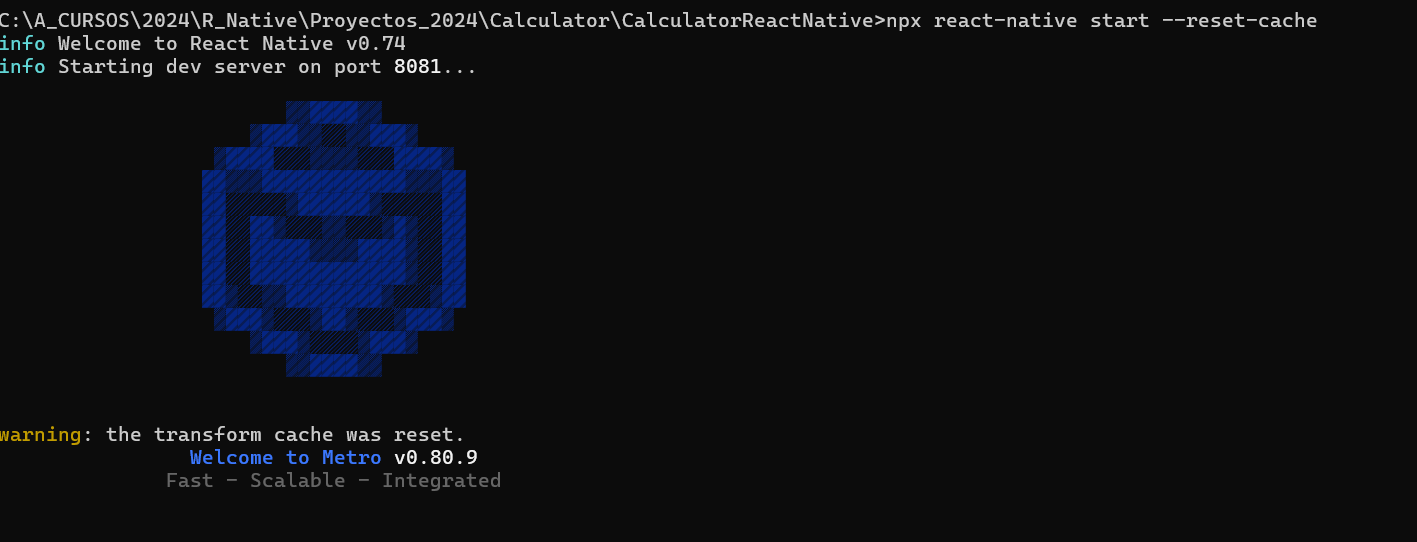
* Limpiar cache y levantar

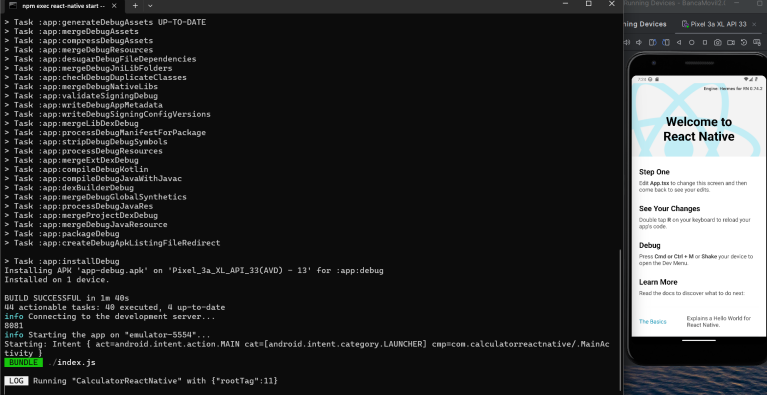
...\CalculatorReactNative> cd android

..\CalculatorReactNative\android>gradlew clean

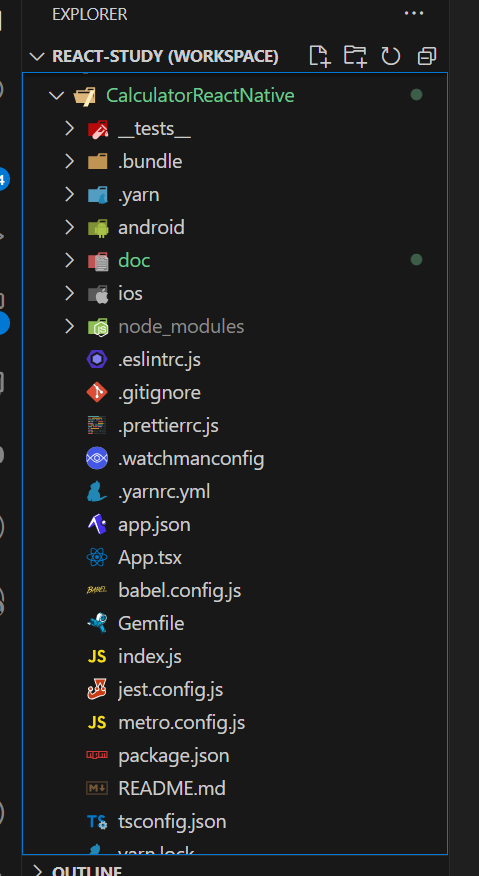


\CalculatorReactNative>npx react-native start --reset-cache





* Abrir en VSCODE, limpiar e iniciar programa



* Código:
  + Componente principal Calculator.jsx donde se despliega la APP y se realiza los cálculos con el apoyo de utilitarios y tipos.
  + Se crearon cuatro componentes
    - CurrentDisplay: Texto de información actual para los cálculos
    - ResultDisplay:Texto de información de entrada de data y resultados
    - BottonMap: Componente que permite recorrer una matriz con los datos de los botones y sus caracteristicas
    - BottonItem: Componente que despliega los botones en la vista final
* Manejo de tipos:

/\* eslint-disable prettier/prettier \*/

export type BotonDataProps = {

  backgroundColorNumber: string;

  value: string;

};

export type ButtonActionsProps = {

  press: (text: string) => void;

};

export type BotonProps = BotonDataProps & ButtonActionsProps;

* Manejo de estilos de la vista
  + Pantalla principal: Se divide en dos parte superior e inferior

const styles = StyleSheet.create({

  container: {

    flex: 1,

    padding: 30,

    backgroundColor: '#f0ebeb',

  },

  titleContainer: {flex: 1, justifyContent: 'center', alignItems: 'center'},

  title: {fontSize: 24, fontWeight: '900', color: '#FFF'},

  northContainer: {flex: 1, backgroundColor: '#000'},

  sourthContainer: {

    flex: 2,

    backgroundColor: '#202020',

  },

});

* + Botones: Se define un tamaño y separación y borde redondeado por botón

const styles = StyleSheet.create({

  constainer: {

    flex: 1,

    paddingTop: 10,

  },

  botonContainer: {

    borderRadius: 20,

    width: 65,

    height: 65,

    justifyContent: 'center',

    alignItems: 'center',

    marginVertical: 10,

    marginHorizontal: 10,

  },

  number: {

    color: '#d8d3d3',

    fontSize: 26,

  },

});

* + Textos de información: SE alinea en la parte inferior y a la derecha de su posición

const styles = StyleSheet.create({

  container: {flex: 1, justifyContent: 'flex-end', alignItems: 'flex-end'},

  text: {fontSize: 50, color: '#FFF'},

});

* Salida: Cálculos y validaciones