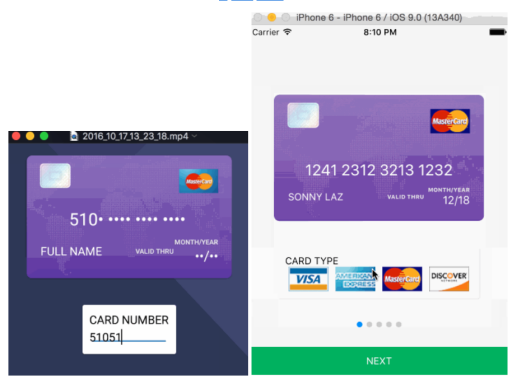
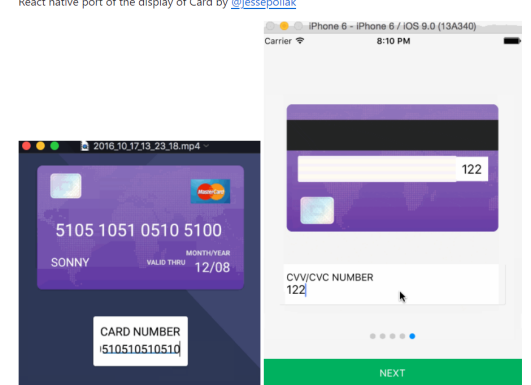
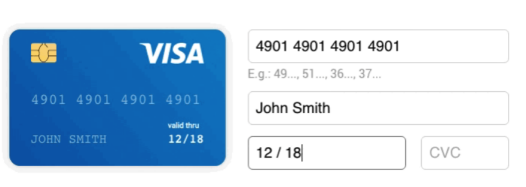
**REACT NATIVE TARJETA DE CREDITO**

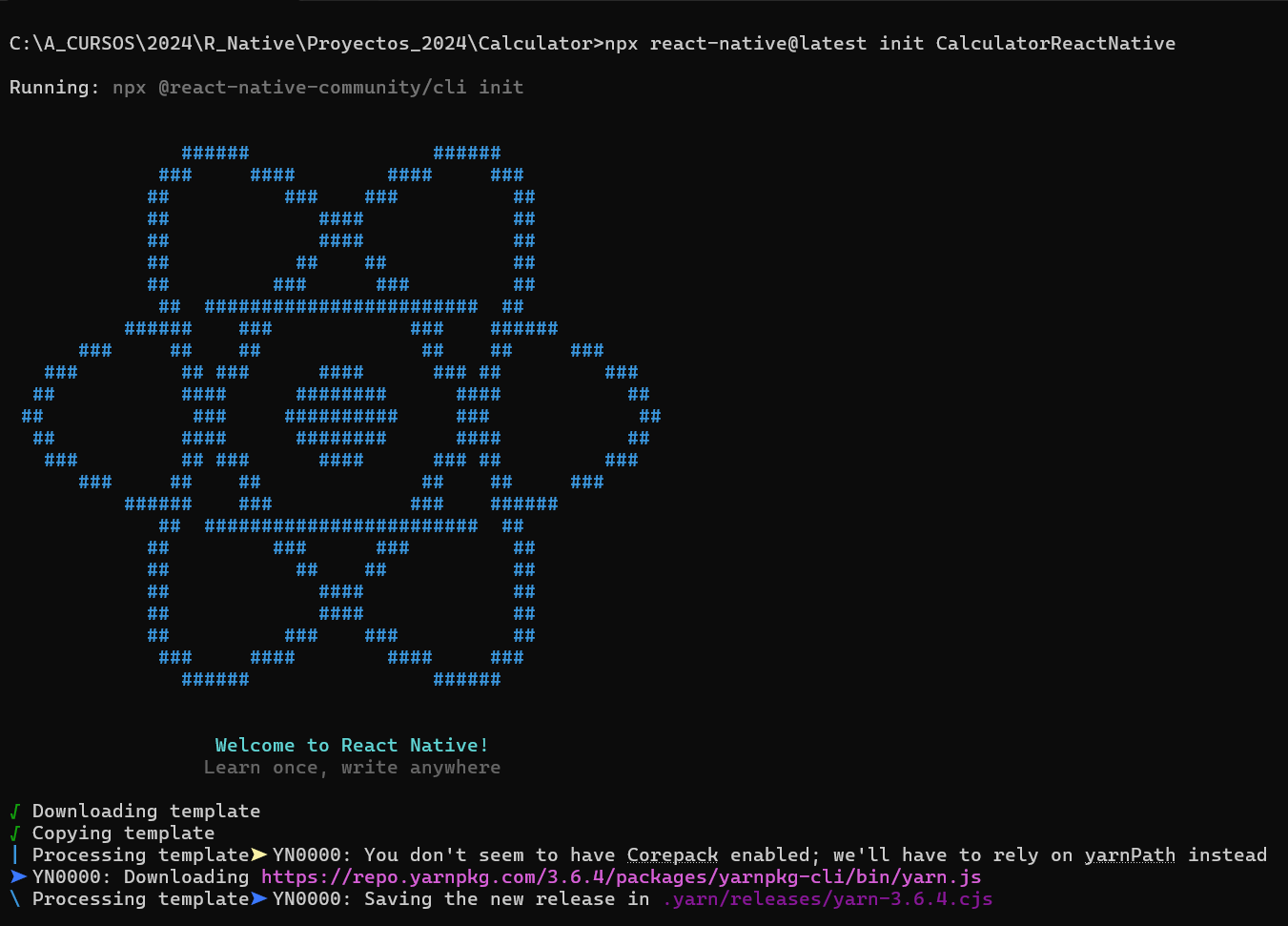
**Proyecto de TDC en React Native: Modelo**

* Creamos un proyecto React Native con TypeScript básico

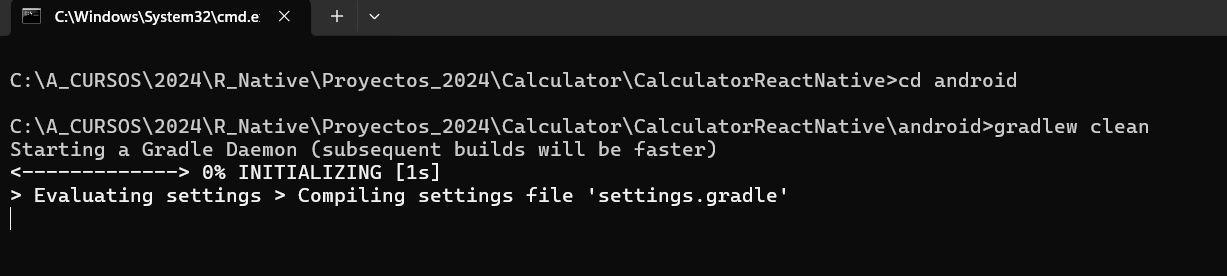
\_> npx react-native@latest init TDCReactNative



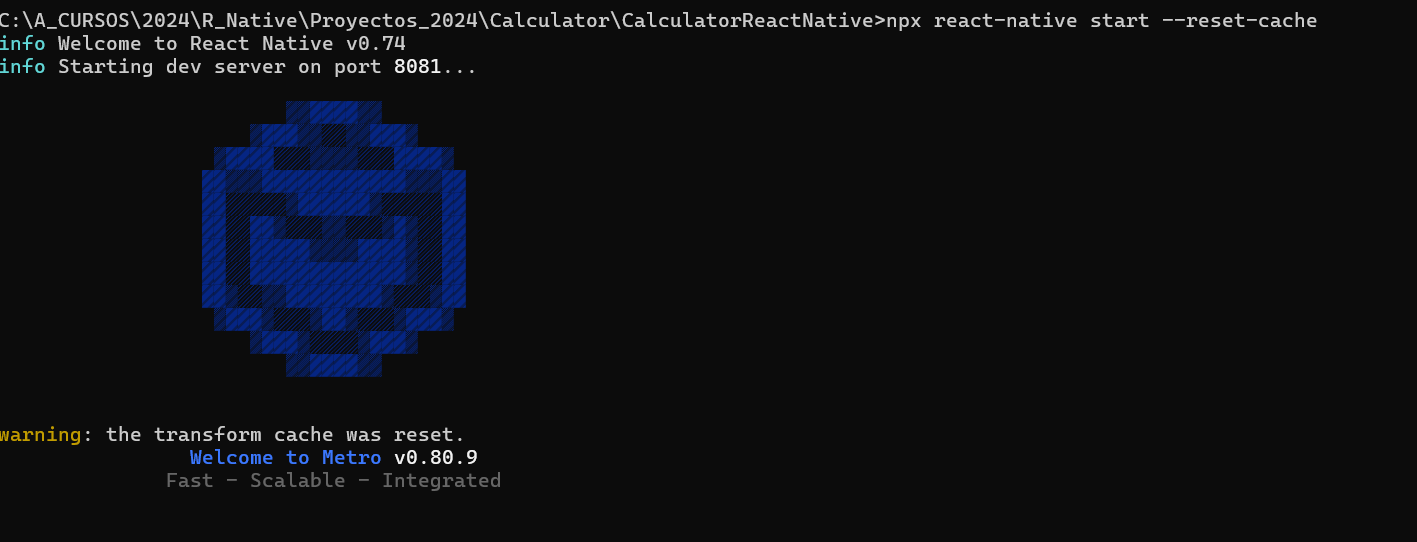
* Limpiar cache y levantar

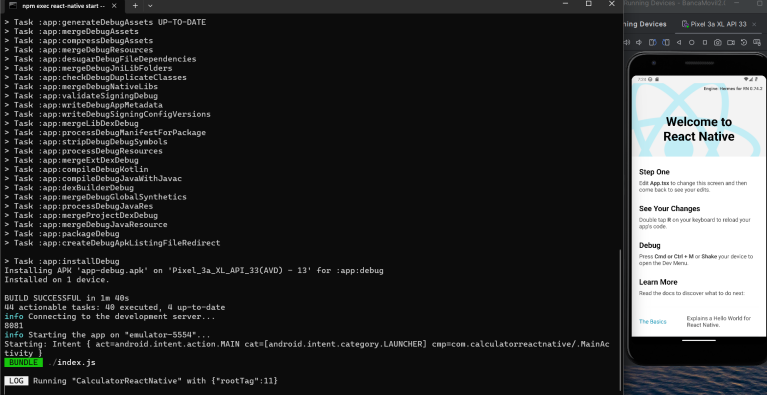
..\CalculatorReactNative> cd android

..\CalculatorReactNative\android>gradlew clean

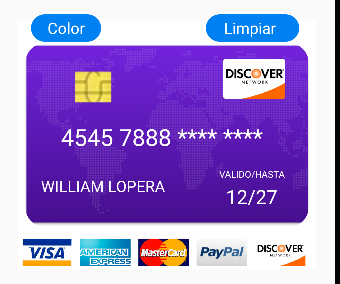


\CalculatorReactNative>npx react-native start --reset-cache





* Abrir en VSCODE, limpiar e iniciar programa
* Código:
  + Componente principal TDC.jsx donde se despliega la APP y se realiza las llamada a los componentes se utilizan estados para controlar los parámetros de los componentes programados.
  + Se crearon cinco componentes
    - TDCDate: Componente para control de mes/año. Validando los campos y fomateando la entrada de datos.
    - TDCImage: Componentes para mostrar front y back de la tardeja de crédito
    - TDCInput: Componente para entrada de datos de número se controla la entrada de números y ‘\*’ permitiendo ir agregando o borrado datos en la medida que se introduce o quitan los valores
    - TDCPicker: Componente para cambiar el color de la trajeta
    - TypeImage: Componente para mostrar las imágenes de logos que se pueden aplicar



* Códigos importantes
  + Método para control de números o ‘\*’

 const formatCardNumber = (number: string) => {

    // Elimina todos los caracteres que no sean dígitos

    const cleaned = ('' + number).replace(/\D/g, '');

    // Aplica el formato: cada 4 dígitos se agrega un espacio

    const match = cleaned.match(/.{1,4}/g);

    // Generar numero formateado a mostrar en la TDC

    const input = number.replaceAll(' ', '');

    let data = input;

    for (let index = 0; index < 16 - input.length; index++) {

      data = data + '\*';

    }

    const cleanedData = data.replaceAll(' ', '');

    const matchData = cleanedData.match(/.{1,4}/g);

    const result = match ? match.join(' ') : number;

    const text = matchData ? matchData.join(' ') : data;

    onChange(result, text);

  };

* + Método para control de fecha (mes/año)

  const formatDate = (input: string) => {

    let data = input.replace('/', '');

    // validar el mes

    if (Number(data.substring(0, 2)) > 12) {

      return;

    }

    if (data.length > 1) {

      data = data.substring(0, 2) + '/' + data.substring(2, data.length + 1);

    }

    if (data.length === 3 && input === data) {

      data = data.replace('/', '');

    }

    onChange(data);

  };

* Imagen back, uso de animación para dar efecto a la TDC back

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

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

import React, {FC, useEffect, useState} from 'react';

import {BackImageProps} from '../../../types';

const BackImage: FC<BackImageProps> = ({url, cvc}) => {

  const fadeAnim = useState(new Animated.Value(0))[0];

  const [showValue, setShowValue] = useState(false);

  useEffect(() => {

    Animated.timing(fadeAnim, {

      toValue: 1,

      duration: 1000,

      useNativeDriver: true,

    }).start(() => {

      setShowValue(true);

    });

  }, [fadeAnim]);

  return (

    <View style={styles.container}>

      <Animated.Image

        source={url}

        style={[styles.image, {opacity: fadeAnim}]}

      />

      {showValue && (

        <View style={styles.cvcView}>

          <Text style={styles.cvc}>{cvc}</Text>

        </View>

      )}

    </View>

  );

};

export default BackImage;

const styles = StyleSheet.create({

  container: {

    flex: 1,

    margin: 10,

    position: 'relative',

  },

  image: {

    width: '100%',

    height: '100%',

    borderRadius: 25,

  },

  cvcView: {

    position: 'absolute',

    top: 95,

    right: 85,

  },

  cvc: {

    color: '#000',

    fontSize: 25,

  },

});

* + Uso de tipos para el tipado de los props

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

import {ImageSourcePropType} from 'react-native';

export type TypeImageItemProps = {

  url: ImageSourcePropType;

};

export type TypeImageActionProps = {

  press: (url: ImageSourcePropType) => void;

};

export type TypeImageProps = TypeImageItemProps & TypeImageActionProps;

export type FrontImageProps = {

  url: ImageSourcePropType;

  urlLogo: ImageSourcePropType;

  number: string;

  name: string;

  date: string;

};

export type TextNumberProps = {

  value: string;

  onChange: (value: string, text: string) => void;

};

export type TextDateProps = {

  value: string;

  onChange: (text: string) => void;

};

export type BackImageProps = {

  url: ImageSourcePropType;

  cvc: string;

};

export type ItemProps = {title: string};

export type ItemSeletedProps = {

  id: string;

  front: ImageSourcePropType;

  back: ImageSourcePropType;

  title: string;

};

export type TDCPickerProps = {

  press: (front: ImageSourcePropType, back: ImageSourcePropType) => void;

};

* Componente principal para notar el manejo de estados y estilos para control del APP

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

import React, {useState} from 'react';

import {

  View,

  ScrollView,

  TextInput,

  StyleSheet,

  ImageSourcePropType,

  Button,

  TouchableOpacity,

  Text,

} from 'react-native';

import FrontImage from '../../components/TDCImage/FrontImage';

import TypeImageMap from '../../components/TypeImage/TypeImageMap';

import TDCInput from '../../components/TDCInput';

import TDCDate from '../../components/TDCDate';

import BackImage from '../../components/TDCImage/BackImage';

import TDCPicker from '../../components/TDCPicker/TDCPicker';

const NUMBER\_BASE = '\*\*\*\* \*\*\*\* \*\*\*\* \*\*\*\*';

const TDC = () => {

  const [urlLogo, setUrlLogo] = useState(

    require('../../assets/logos/master-logo.png'),

  );

  const [number, setNumber] = useState('');

  const [textNumber, setTextNumber] = useState(NUMBER\_BASE);

  const [name, setName] = useState('');

  const [date, setDate] = useState('');

  const [cvc, setCvc] = useState('');

  const [isCVC, setIsCVC] = useState(false);

  const [tdcImage, setTdcImage] = useState({

    front: require('../../assets/tdc/card-black-front.png'),

    back: require('../../assets/tdc/card-black-back.png'),

  });

  const handleLogoPress = (url: ImageSourcePropType) => {

    if (isCVC) {

      return;

    }

    setUrlLogo(url);

  };

  const handleClean = () => {

    setNumber('');

    setTextNumber(NUMBER\_BASE);

    setName('');

    setDate('');

    setCvc('');

  };

  const handleNumber = (value: string, text: string) => {

    setNumber(value);

    setTextNumber(text);

  };

  const handleTDCColor = (

    front: ImageSourcePropType,

    back: ImageSourcePropType,

  ) => {

    setTdcImage({front, back});

  };

  return (

    <View style={styles.container}>

      <View style={styles.nortContainer}>

        <View style={styles.bottons}>

          <View style={styles.picker}>

            <TDCPicker press={handleTDCColor} />

          </View>

          <View style={styles.botton}>

            <TouchableOpacity onPress={handleClean} style={styles.botton}>

              <Text style={styles.bottonText}>Limpiar</Text>

            </TouchableOpacity>

          </View>

        </View>

        {isCVC ? (

          <BackImage cvc={cvc} url={tdcImage.back} />

        ) : (

          <FrontImage

            url={tdcImage.front}

            urlLogo={urlLogo}

            number={textNumber}

            name={name}

            date={date}

          />

        )}

        <TypeImageMap press={handleLogoPress} />

      </View>

      <View style={styles.sourthContainer}>

        <ScrollView>

          <TDCInput value={number} onChange={handleNumber} />

          <TextInput

            style={styles.nameInput}

            placeholder="Nombre del usuario"

            value={name}

            maxLength={20}

            onChangeText={input => setName(input.toUpperCase())}

          />

          <View style={styles.dataView}>

            <TDCDate value={date} onChange={setDate} />

            <TextInput

              style={styles.cvcInput}

              placeholder="CVC"

              value={cvc}

              maxLength={3}

              onChangeText={setCvc}

              keyboardType="numeric"

              onFocus={() => setIsCVC(true)}

              onBlur={() => setIsCVC(false)}

            />

          </View>

        </ScrollView>

      </View>

    </View>

  );

};

const styles = StyleSheet.create({

  container: {

    flex: 1,

  },

  nortContainer: {

    height: 320,

    backgroundColor: '#FFF',

  },

  bottons: {

    height: 35,

    margin: -8,

    flexDirection: 'row',

  },

  picker: {

    flex: 1,

    alignItems: 'flex-start',

    marginLeft: 25,

    zIndex: 10,

  },

  botton: {

    backgroundColor: '#0081F1',

    height: '100%',

    width: 90,

    marginRight: 30,

    justifyContent: 'center',

    alignItems: 'flex-end',

    borderRadius: 25,

  },

  bottonText: {

    fontSize: 20,

    color: '#FFF',

  },

  image: {

    width: '100%',

    height: '100%',

    borderRadius: 25,

    resizeMode: 'contain',

  },

  sourthContainer: {

    flex: 1,

    marginBottom: 10,

    paddingTop: 20,

    paddingBottom: 20,

  },

  nameInput: {

    borderRadius: 4,

    padding: 2,

    marginVertical: 8,

    marginHorizontal: 10,

    width: '95%',

    borderWidth: 1,

    borderColor: '#a19e9e',

    fontSize: 24,

    color: '#000',

  },

  cvcInput: {

    borderRadius: 4,

    padding: 2,

    marginVertical: 8,

    marginHorizontal: 10,

    width: '45%',

    borderWidth: 1,

    borderColor: '#a19e9e',

    fontSize: 24,

    color: '#000',

  },

  dataView: {flexDirection: 'row'},

});

export default TDC;

* Salida:

