UNIVERSIDAD PRIVADA FRANZ TAMAYO
FACULTAD DE INGENIERÍA
CARRERA DE SISTEMAS

“DEFENSA HITO 3 - TAREA FINAL”

Nombre Completo: Isaac Limbert Herrera Mareño

Asignatura: PDM

Carrera: INGENIERÍA DE SISTEMAS

Paralelo: PDM

Docente: Lic. William R. Barra Paredes

Fecha: 11/05/2020

GitHub: https://github.com/isaac0155/PDM-Isaac-Herrera-MAre-o

1. Preguntas.

Responda de manera breve y clara posible.

I. Defina que es un componente en React Native y muestre un ejemplo.

Un componente es un elemento de software visual que tiene su propio estado, recibe unas propiedades e implementa su propia lógica de renderizado.

Aceptan entradas arbitrarias (llamadas “props”) y devuelven a React elementos que describen lo que debe aparecer en la pantalla.

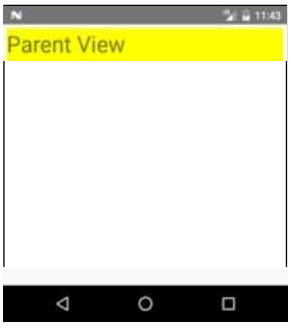
Un componente es un elemento muy básico en react-native. Podemos dividir la aplicación grande en muchos componentes pequeños. Esto hace que el desarrollo sea rápido y mantiene el código muy claro de entender.

En React, View es un componente integrado. Si está familiarizado con div en HTML, la vista es como div, se usa en aplicaciones móviles. La vista es un área de contenido donde muestra su contenido. Con esto, puede organizar el contenido de una manera muy buena.

Por ejemplo:

```
render () {

  regreso (
    < View estilo = {{backgroundColor: ”#541284”, margen: 5}}>
        <Text> Parent View </Text
    </View>
  );
 }
}
AppRegistry.registerComponent ('NativeSample', () => NativeSample);
```



II. Explique cómo se realiza la navegación entre screens en React Native.

Para realizar una navegación entre Screens en react-native primero se diseñan los botones, uno para navegar al siguiente screen y otro para el anterior, utilizamos el atributo **onPress** de cada botón para navegar entre Screens.

Luego importamos esos botones en un Screen, luego llamamos a los botones y le damos valor al atributo **onPressPrev** Y **onPressNext** para navegar entre Screens.

```
export default class ButtonFooter extends Component
{
  constructor(props)
  {
    super(props);
  }
  render()
  {
    return(
        <View>
            <Buton titleButton="Siguente"

onPress={this.props.onPressPrev}
            />
            <Buton titleButton="Anterior"

onPress={this.props.onPressNext}
            />
        </View>
    );
  }
}
```

```
import ButtonFooter from '.
/ButtonFooter'; const AboutScreen =
({navigation}) =>{ return(
      <View style={this.props.stilo} >
        <ButtonFooter
        style={{marginTop:100}}
        onPressPrev={()=> navigation.navigate('ScreenAnterior')}
onPressNext={()=> navigation.navigate('ScreenSiguiente')}
        />
      </View>
      );
}
```

III. Que significa IaaS, PaaS y SaaS .

La computación en la nube dio origen a muchos acrónimos, pero ninguno es más importante que IaaS, PaaS y SaaS.

Son infraestructuras como servicio (IaaS), plataforma como servicio (PaaS) y software como servicio (SaaS).

IaaS

La infraestructura como servicio (IaaS) es la más simple de las tres categorías para definir, ya que es ampliamente el mismo independientemente del proveedor que elija. En pocas palabras, IaaS es un tercero que proporciona una infraestructura de TI altamente automatizada y escalable (almacenamiento, alojamiento, computación, redes) y solo cargos por lo que usa.

Por lo tanto, en lugar de poseer activos como licencias de software o servidores, las empresas pueden alquilar recursos de forma flexible de acuerdo con sus necesidades.

PaaS

La plataforma como servicio (PaaS) es posiblemente el más difícil de los tres modelos de nubes para definir limpiamente. La idea es proporcionar todos los conceptos básicos de IaaS, así como las herramientas y capacidades necesarias para desarrollar e implementar

aplicaciones de forma segura. Eso podría ser middleware, administración de bases de datos, análisis o un sistema operativo.

Una plataforma como servicio debería proporcionarle a un desarrollador todo lo que necesita para construir e implementar una aplicación sin tener que hacer ninguna provisión de la infraestructura subyacente.

Los proveedores de PaaS tienden a ser las mayores empresas de tecnología, que pueden ofrecer una amplia gama de capacidades para sus clientes en una plataforma. Los ejemplos incluyen Google App Engine, Oracle Cloud Platform, Pivotal's Cloud Foundry y Heroku, propiedad de Salesforce.

SaaS

El software como servicio (SaaS) es el lugar donde una parte del software está alojada por un tercero y se puede acceder a través de la web, normalmente solo iniciando sesión, y generalmente se cobra por suscripción o por usuario. Esto difiere del antiguo modelo de compra e instalación de software en una máquina o servidor de forma manual.

SaaS es mucho más relevante para aplicaciones muy específicas, como el correo electrónico o el software de gestión de relaciones con el cliente (CRM). Cualquiera que haya usado una aplicación de Google como Gmail o Google Docs, o almacenamiento de archivos en la nube como Dropbox, habrá usado una pieza de SaaS.

IV. Que es Firebase, Firestore y explique a que se refiere cuando se habla de Baas.

Firebase

Firebase es la nueva y mejorada plataforma de desarrollo móvil en la nube de Google. Se trata de una plataforma disponible para diferentes plataformas (Android, iOS, web), con lo que de esta forma presentan una alternativa seria a otras opciones para ahorro de tiempo en el desarrollo como Xamarin.

Con Firebase, podemos:

- lograr una integración dinámica de los usuarios usando **Firebase Authentication**;
- lograr que nuestras aplicaciones sean visualizadas y utilizadas utilizando la herramienta de **compartir** o **Dynamic Links**;
- enviar notificaciones a varias plataformas con **Cloud Messaging**;
- crear análisis de resultados con **Analytics**;

Firestore

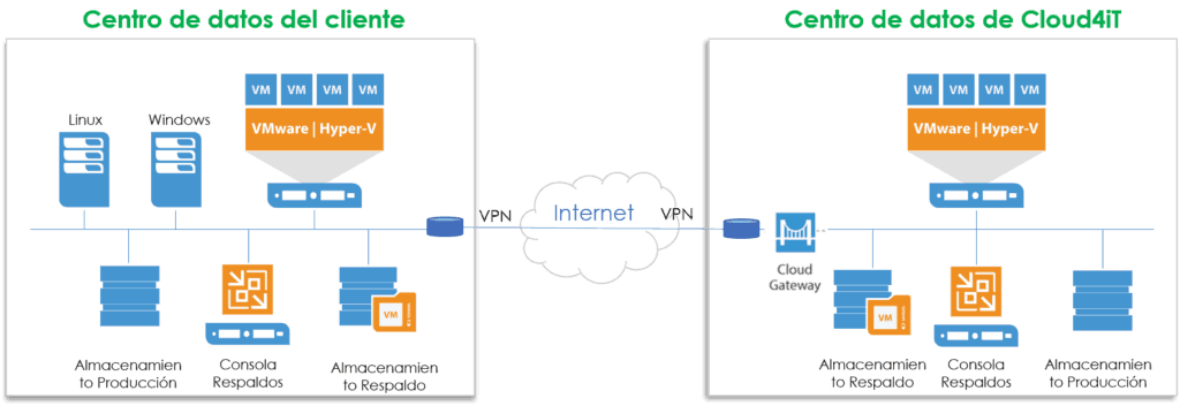
Cloud Firestore es una base de datos flexible y escalable para la programación en servidores, dispositivos móviles y la Web desde Firebase y Google Cloud Platform.

Al igual que Firebase Realtime Database, mantiene tus datos sincronizados entre apps cliente a través de agentes de escucha en tiempo real y ofrece asistencia sin conexión para dispositivos móviles y la Web, por lo que puedes compilar apps con capacidad de respuesta que funcionan sin importar la latencia de la red ni la conectividad a Internet.

Cloud Firestore también ofrece una integración sin interrupciones con otros productos de Firebase y Google Cloud Platform, incluido Cloud Functions.

Baas

En la siguiente imagen te mostramos como funciona una de las soluciones de BaaS, utilizando el software de **VEEAM.**



Del lado izquierdo vemos los clientes o “tenants” que tienen su propia infraestructura virtual (Vmware o Hyper-V) en el caso de Veeam software. En dicha infraestructura se encuentra el backup server y este servidor realiza copias de resguardo locales que son replicadas al Service Provider (SP) por medio del Cloud Gateway que es un servicio que ofrece el fabricante de software para asegurar dos temas:

1. Que la conexión por la que viaja la información este encriptada con certificados de seguridad SSL

2. Que el lugar a donde llegue el respaldo sea certificado por el fabricante. En este caso por Veeam.

Como verás del lado del service provider o proveedor de servicios (Cloud4iT) también hay un backup server y disco suficiente para almacenar los respaldos (Cloud Repository).

La idea es que con la consola de respaldo del cliente sea posible generar los respaldos, pero también reestablecer la información en caso necesario.

V. Defina o explique si React es lo mismo que Reac Native. Si son distintos liste cuales son las diferencias.

ReactJs

ReactJS es una librería JavaScript, desarrollada y mantenida por Facebook, enfocada a la visualización. Como React se encarga sólo de la vista de la aplicación dentro de un paradigma Modelo-Vista-Controlador, desde Facebook recomiendan encarecidamente el uso de ReactJS con Flux, que es grosso modo, un patrón de diseño software. Por este motivo, muchas veces vemos en combinación a React JS y Redux. Esta es una librería que implementa este patrón de diseño Flux y que ocupa apenas 2Kb.

ReactJS está fuertemente basado en componentes. Estos componentes son los elementos que constituyen la interfaz del usuario (un botón, un buscador, etc).

Reaccionan (de ahí viene su nombre) tanto a los eventos producidos por el usuario como a los del servidor. Finalmente, estos se repintan a sí mismos cuando ocurre un cambio de estado.

Con cada alteración se modifica un DOM virtual. Este hecho hace que el DOM real solo cambie en las partes que han experimentado alteraciones. Esto se traduce en una mejora del consumo de memoria y el rendimiento con respecto a otros frameworks.

React Native

Antes de entrar en más detalle sobre React Native, hay que tener muy presente la diferencia entre dos tipos de tecnologías de cara al desarrollo de aplicaciones móviles: Cross-platform (o híbridas) y Nativa.

- Las apps híbridas se desarollan usando HTML5, CSS y JavaScript. Es decir, utilizan el mismo código independientemente de la plataforma en que se ejecutan. Se incrustan dentro de una webview o una Web App.

- Las apps nativas se desarrollan usando el lenguaje requerido por la plataforma de destino en concreto: Objective-C y Swift para iOS, Java y Kotlin para Android, etc.

Estos dos enfoques han imperado durante mucho tiempo en el desarrollo mobile. Pero con la irrupción de React Native este enfoque está cambiando. React Native es un mobile framework JavaScript que usa los componentes de ReactJS para construir aplicaciones “nativas” para móviles. Es decir, su desarrollo no es 100% cross-platform. Y es aquí donde radica su importancia frente a otros frameworks mobile y el porqué está revolucionando el desarrollo móvil.

Las vistas las crea de forma nativa. Esto significa que tiene un intérprete que a medida que la app se va ejecutando. Lo lee el HTML de estas vistas y va colocando cada uno de los componentes nativos en su posición.

Por ejemplo, el caso en el que la vista expresa que hay que poner un botón en la pantalla Home.

Cuando la app llegue a esta pantalla React Native creará un botón nativo y lo colocará en su posición correspondiente. Esto le dará una clara ventaja de que todo el frontend. Es decir, todas las interacciones con el usuario serán nativas.

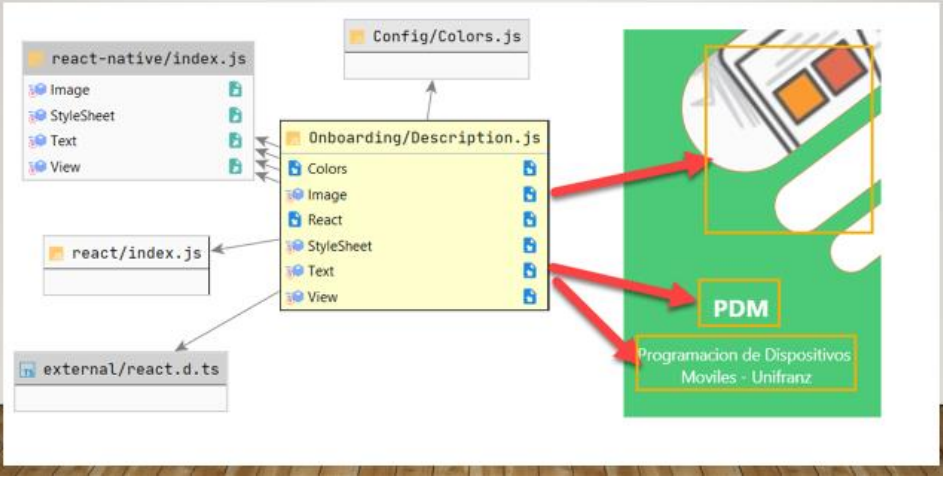
PARTE PRACTICA

```
import React, {Component} from 'react';
import {StyleSheet, View, Text,TouchableOpacity} from 'react-native';
import Colores from '../Config/Colores';

export default class BotonDinamico extends Component{
  constructor(props){
  super(props);
  }
  render(){
    return(
      <View>
        <TouchableOpacity style={buttonStyle.pinta } onPress={this.props.onPress}>
          <Text style={buttonStyle.text}>{this.props.titleButton}</Text>
        </TouchableOpacity>
      </View>
    );
  }
}
const buttonStyle = StyleSheet.create({
  text: {
    textAlign: 'center',
    color: Colores.blanco,
    height: 10,
  },
  pinta: {
    marginBottom: 10,
    width: 70,
    paddingVertical: 10,
    borderRadius: 20,
    width: '100%',
    justifyContent: 'center',
    borderWidth: StyleSheet.hairlineWidth,
    backgroundColor: 'transparent',
  },
});
```

- Primero se importa los componentes que se vana utilizar Colors, React, StyleSheet
- Se declara el constructor, listo para exportar la Clase
- Se utiliza view como un contenedor de componentes
- Luego declaramos el Button y le damos el respectivo texto al botón
- Al botón de damos el atributo OnPress para realizar la respectiva acción cuendo de presione al boton
- Declaramos StyleSheet dándole un nombre y le damos los atributos para darle estilos y formas a los componentes

PREGUNTA 2
CREAR EL COMPONENTE DESCRIPTION.JS



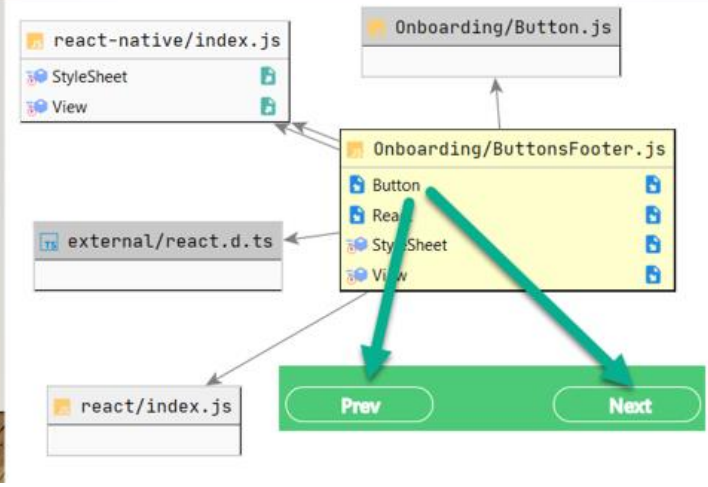

```
import React, {Component} from 'react'
import {StyleSheet,View,Text,Image} from 'react-native';
export default class Descripcion extends Component{
constructor(props){
  super(props);
  }
  render(){
    return(
      <View style={estilos.grande}>
        <Image source={this.props.source} style={estilos.imagen}/>
        <Text style={estilos.title}>{this.props.letrasPrimero}</Text>
      </View>
      <View style={estilos.grande}>
        <Text style={estilos.letras}>{this.props.letrasSegundo}</Text>
        <Text style={estilos.letras}>{this.props.letrasTercero}</Text>
      </View>
    );
  }
}
const estilos = StyleSheet.create({
  title:{
    fontWeight:'bold',
    backgroundColor:'transparent',
    fontSize:30 ,
    color:'white',
    marginTop:40,
  },
  grande:{
    alignItems:'center',
    justifyContent:'center',
  },
  letras:{
    color:'white',
    backgroundColor:'transparent',
    marginTop:40,
    fontWeight:'bold',
  },
  imagen:{
    width:300,
    height:300,
  }
```

- Primero se importa los componentes que se vana utilizar Colors, React, StyleSheet

- Se declara el constructor, listo para exportar la Clase
- Se utiliza view como un contenedor de componentes
- Utilizamos el tag para usar una imagen y con el atributo source, enviamosla imagen que queremos que se muestre
- Luego acomodamos los textos y con This.props.Letras recibimos el contenido del texto

- Declaramos StyleSheet dándole un nombre y le damos los atributos para darle estilos y formas a los componentes

PREGUNTA 3
CREAR EL COMPONENTE BUTTONSFOTEER.JS

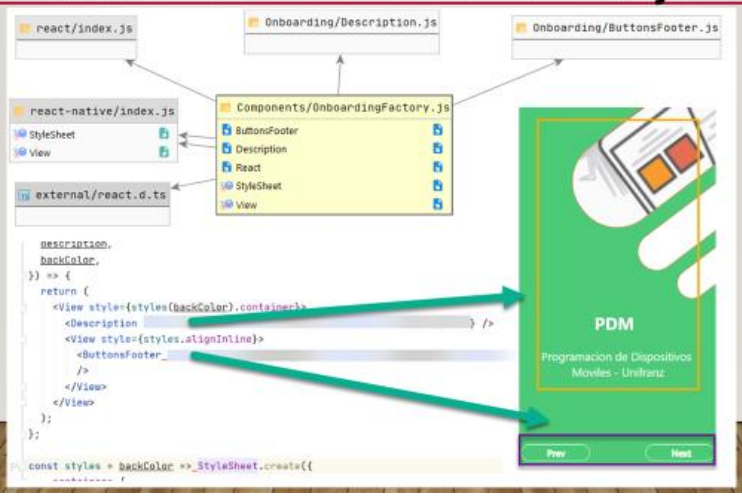



```
import React, {Component} from 'react';
import {StyleSheet, View, Text,TouchableOpacity} from 'react-native';
import Colores from '../../Colores';
import Buton from './BotonDinamico';
import { styles } from 'expo-ui-kit';
export default class ButtonFooter extends Component{
  constructor(props){
  super(props);
  }
  render(){
    return(
      <View style={buttonStyle.row}>
        <Buton titleButton={this.props.Prev} onPress={this.props.onPressPrev}/>
        <Buton titleButton={this.props.Next} onPress={this.props.onPressNext}/>
      </View>
    );
  }
}
const buttonStyle = StyleSheet.create({
  container: {
    borderWidth: StyleSheet.hairlineWidth,
    justifyContent: 'center',
    paddingVertical: 10,
    marginBottom: 10,
    width: '100%',
    backgroundColor: 'transparent',
    borderRadius: 20,
    width: 70,
  },
  text: {
    color: Colores.blanco,
    textAlign: 'center',
    height: 20,
  },
  row:{
    flex:1,
    flexDirection:'row',
    justifyContent:'space-between',
    marginBottom:10,
  },
});
```

- Primero se importa los componentes que se vana utilizar Colors, React, StyleSheet
- Luego importamos el Boton que creamos en la otra clase, para usarlo en este
- Se declara el constructor, listo para exportar la Clase
- Se utiliza view como un contenedor de componentes
- A los botónes le damos el atributo OnPress para realizar la respectiva acción cuendo de presione al botón ya se Prev o Next

- Declaramos StyleSheet dándole un nombre y le damos los atributos para darle estilos y formas a los componentes

PREGUNTA 4
CREAR EL COMPONENTE ONBOARDINGFACTORY.JS

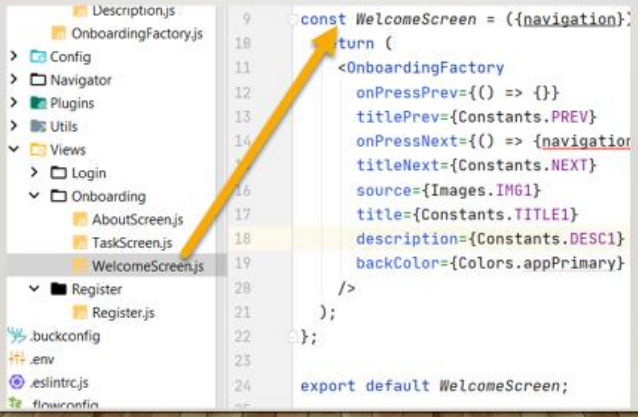



```
import React, {useState,Component} from 'react';
import { StyleSheet, View, SafeAreaView, KeyboardAvoidingView, Alert } from 'react-native';
import Colores from '../Colores';
import BotonDinamico from '../Componentes/BotonDinamico';
import Description from '../Componentes/Description';
import{NavigationContainer} from '@react-navigation/native';
import {createStackNavigator} from '@react-navigation/stack';
const _onPressPrev = () => {console.log('Anterior')};
const _onPressNext = () => {console.log('Siguiente')};
export default class OnboardingFactory extends Component{
  constructor(props){
  super(props);
  }
  render(){
    return(
      <View style={this.props.stilo} >
        <View style={{marginTop:100}}>
          <Description source={this.props.source} letrasPrimero={this.props.letrasPrimero}
            letrasSegundo={this.props.letrasSegundo} letrasTercero={this.props.letrasTercero}/>
          <BotonDinamico style={{marginTop:100}} Next={this.props.Next} onPressNext={this.props.onPressNext}
            Prev={this.props.Prev} onPressPrev={this.props.onPressPrev}/>
        </View>
      </View>
    );
  }
}
```

- Primero se importa los componentes que se vana utilizar Colors, React, StyleSheet
- Luego importamos el Boton que creamos en la otra clase, para usarlo en este
- Luego importamos la clase Descripcion para usar en esta clase
- Tambien importamos en container de React

- Se declara el constructor, listo para exportar la Clase
- Se utiliza view como un contenedor de componentes
- Le damos el valor a sus atributos a la clase instanciada Description
- Le damos el valor a sus atributos a la clase instanciada BotonDinamico

PREGUNTA 5
CREAR LOS SCREENS PARA EL NAVIGATOR Y VERIFICAR EL LOGIN A FIREBASE.



SE PIDE CREAR 3 SCREENS, EMPECEMOS CON ABOUTSCREEN

```
import React, {useState} from 'react';
import { StyleSheet, View, SafeAreaView, KeyboardAvoidingView, Alert } from 'react-native';
import Images from '../images';
import Colores from '../Colores';
import OnboardingFactory from '../../Components/Onboarding/OnboardingFactory';
const AboutScreen = ({navigation}) => {
  return(
    <OnboardingFactory stilo={estilos.contenido}
      source={Images.SCREEN2} Next={'Anterior'} Prev={'Siguiente'}
      letrasPrimero={'Defensa Hito 3'} letrasSegundo={'Isaac Limbert Herrera Mareño'}
      letrasTercero={'Gestion 2020'} onPressNext={()=> navigation.navigate('TaskScreen')}
      onPressPrev={()=> navigation.navigate('WelcomeScreen')} >
    </OnboardingFactory>
  );
};
const estilos = StyleSheet.create({
  contenido: {
    backgroundColor: Colores.naranja,
    justifyContent: 'center',
    alignItems: 'center',
  },
});
export default AboutScreen;
```

- Primero se importa los componentes que se vana utilizar Colors, React, StyleSheet, React, Imagenes
- Luego importamos la clase OnboardingFactory para usar en esta clase
- Se declara el constructor, listo para exportar la Clase
- Instanciamos OnboardingFactory y le damos sus atributos de estilos
- Damos valor a los campor de texto que esta clase nos pide y se la enviamos
- Declaramos StyleSheet dándole un nombre y le damos los atributos para darle estilos y formas a los componentes
- Por ultimo Exportamos este Screen.

SE PIDE CREAR 3 SCREENS, EMPECEMOS CON TASKSCREEN

```jsx
import React, {useState} from 'react';
import { StyleSheet, View, SafeAreaView, KeyboardAvoidingView, Alert } from 'react-native';
import Images from '../images';
import Colores from '../Colores';
import OnboardingFactory from '../../Components/Onboarding/OnboardingFactory';
const TaskScreen = ({navigation}) => {
  return(
    <OnboardingFactory stilo={estilos.contenido} source={Images.SCREEN3}
      Next={'Anterior'} Prev={'Siguiente'} letrasPrimero={'FIREBASE'}
      letrasSegundo={'integracion de React-native'} letrasTercero={'con Firebase'}
      onPressPrev={()=> navigation.navigate('AboutScreen')}
      onPressNext={()=> navigation.navigate('login')} >
    </OnboardingFactory>
  );
};
const estilos = StyleSheet.create({
  contenido: {
    backgroundColor: Colores.blanco,
    justifyContent: 'center',
    alignItems: 'center',
  },
});
export default TaskScreen;
```

- Primero se importa los componentes que se vana utilizar Colors, React, StyleSheet, React, Imagenes
- Luego importamos la clase OnboardingFactory para usar en esta clase
- Se declara el constructor, listo para exportar la Clase
- Instanciamos OnboardingFactory y le damos sus atributos de estilos
- Damos valor a los campor de texto que esta clase nos pide y se la enviamos
- Declaramos StyleSheet dándole un nombre y le damos los atributos para darle estilos y formas a los componentes
- Por ultimo Exportamos este Screen.

SE PIDE CREAR 3 SCREENS, EMPECEMOS CON WELCOMESCREEN

```jsx
import React, {useState} from 'react';
import { StyleSheet, View, SafeAreaView, KeyboardAvoidingView, Alert } from 'react-native';
import Images from '../img';
import Colores from '../Colores';
import OnboardingFactory from '../../Components/Onboarding/OnboardingFactory';
const WelcomeScreen = ({navigation}) => {
  return(
    <OnboardingFactory stilo={setilos.contenido} source={Images.SCREEN1}
      Next={'Anterior'} Prev={'Siguiente'} letrasPrimero={'POM'}
      letrasSegundo={'Progamacion de Dispositivos Moviles'} letrasTercero={'Unifranz'}
      onPressNext={()=> navigation.navigate('AboutScreen')} >
    </OnboardingFactory>
  );
};
const setilos = StyleSheet.create({
  contenido: {
    backgroundColor: Colores.red,
    justifyContent: 'center',
    alignItems: 'center',
  },
});
export defau lt WelcomeScreen;
```

- Primero se importa los componentes que se vana utilizar Colors, React, StyleSheet, React, Imagenes
- Luego importamos la clase OnboardingFactory para usar en esta clase
- Se declara el constructor, listo para exportar la Clase
- Instanciamos OnboardingFactory y le damos sus atributos de estilos
- Damos valor a los campor de texto que esta clase nos pide y se la enviamos
- Declaramos StyleSheet dándole un nombre y le damos los atributos para darle estilos y formas a los componentes
- Por ultimo Exportamos este Screen.

PANTALLA DE LOGUEO EXITOSO

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

import Constantes from '../contantes';
import Colores from '../Colores';
import ButtonLogin from '../../Components/login/Button';

const RegisterScreen = ({navigation}) => {
  const onPress = () => {
  console.log('register');
  };
  return (
    <View styles={estilos.contenido}>
      <Text>Registro Completado</Text>
      <ButtonLogin onPress={onPress}
        titleButton={Constantes.STRING.REGISTER} />
    </View>
  );
};
const estilos = StyleSheet.create({
  contenido: {
    flex: 1,
    backgroundColor: Colores.azul,
    alignItems: 'center',
  },
});
export default RegisterScreen;
```

- Primero se importa los componentes que se vana utilizar Colors, React, StyleSheet, React
- Luego importamos la clase ButtonLogin para usar en esta clase

- Se declara el constructor, listo para exportar la Clase
- Utilizamos View como contenedor de otros componentes
- Instaciamos ButtonLogin y le mandamos los atributos que necesita, tanto como su texto

- Declaramos StyleSheet dándole un nombre y le damos los atributos para darle estilos y formas a los componentes
- Por ultimo Exportamos este Screen.

Vista de Pantallas finales