Universidad Tecnológica de Puebla
Ingeniería en desarrollo y gestión de software

Desarrollo Móvil Integral

Profesor: Vázquez Mora Paulo Daniel

Producto 2: Componentes funcionales para prototipado de App móvil de Becas

Fecha de entrega: 20 de noviembre de 2022

Grupo: 10mo C

Integrantes:

- Romero Ramírez Miriam
- Romero Titla Anais
- Amaro Galindo Cecilia
- Ramírez Huerta Juan
- Martínez Crisanto Araceli

Periodo: septiembre – diciembre 2022

En el presente documento se mostrará el código realizado para el diseño de interfaces del “Sistema Integral de Información de la Universidad Tecnológica de Puebla”. Se crearon dos pantallas: Inicio/Presentación e Inicio de Sesión.

```
const Stack = createStackNavigator();

const StackNavigator = () => {
  return (
    <NavigationContainer>
      <Stack.Navigator>
        <Stack.Screen
          name = "Index"
          options = {{ title: "Bienvenida"}}
          component = {Inicio}
          />
        <Stack.Screen
          name = "Login"
          options = {{ title: "Inicio de sesión"}}
          component = {Login}
        />
      </Stack.Navigator>
    </NavigationContainer>
  )
}
```

Ilustración 1. Código de componente: StackNavigator.js

Para iniciar se configuro el Stack Navigator en la carpeta src para guardarlo como un componente que se importara en App.js. Se utilizaron las importaciones de “NavigationContainer” y “createStackNavigator”, incluyendo las screens de “Login” e “Inicio”.
La ilustración 1, muestra como se realizo un contenedor que contendría los stacks. Estos stacks hacen referencia a las pantallas importadas, ambas contendrán un titulo en sus propiedades, tanto “Bienvenida” como “Inicio de Sesión”. Con este componente se podrá viajar entre la pantalla de Bienvenida a la pantalla de Inicio de Sesión.

Código de componente: Button

```
const Button = ({
    navigation,
    href,
    text = 'Botón sin editar',
    color = '#089020',
}) => {
  return (
    <TouchableOpacity
      style = {{
        backgroundColor: color,
        padding: 10,
        alignItems: "center",
        justifyContent: "center",
        marginVertical: 1,
        borderRadius: 50,
        width: '100%',
        padding: 15,

      }}
      onPress = { () => {
          navigation.navigate(href)
        }}
      >
        <Text style ={{fontSize: 15, color: 'white'}}>
          {text}
        </Text>
      </TouchableOpacity>
  );
}
```

Ilustración 2. Código de componente: Button.js

Para este componente, se declararon propiedades:

navigation	necesario para viajar entre pantallas por el método onPress
href	recibe la referencia del nombre de la pantalla a la cual se viajará
text	editable para cambiarlo según se esté utilizando
color	el color de fondo que llevara al mostrarse el botón

Este componente se realizó a través de un TouchableOpacity que recibe las propiedades y, además, este ya tiene sus propiedades de estilos definidas y no todas pueden ser editadas a excepción del fondo. En este se declarará el onPress que recibirá el método y la referencia para poder desplazarse entre la pantalla de Bienvenida a Inicio de sesión. Para mostrar el texto que se desea en cada botón, únicamente se utiliza un componente del tipo Text predeterminado de React. Resultado:

Código de componente: InputText

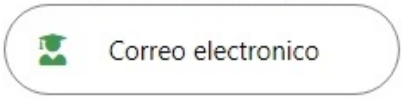
```
const InputText = ({
    secureTextEntry = false,
    placeholder = 'Caja de texto sin editar',
    name = 'lock'
}) => {
  return (
    <View style = {styles.principal}>
        <View style = {styles.icono}>
         <FontAwesome5 name={name} size={18} color="#409850" />
        </View>
      <View style = {styles.contenedor}>
        <TextInput
          secureTextEntry = {secureTextEntry}
          keyboardType = 'email-address'
          placeholder = {placeholder}
          style = {{
            justifyContent: 'center',
            paddingTop: 15,
            paddingRight: 15,
            paddingBottom: 15,
            paddingLeft: 5,
            fontSize: 15,
            borderBottomEndRadius: 50,
            borderTopEndRadius: 50
          }}
        />
      </View>
    </View>
  )
}
```

Ilustración 3. Código de componente: InputText.js

Para este componente se declararon las siguientes propiedades:

secureTextEntry	Necesario para determinar que tipo de texto entrara, es decir, si este pertenece a una contraseña o no.
placeholder	El texto que se mostrara dentro del componente.
name	El nombre del icono que se mostrara en el TextInput.

Este componente está formado a través de la etiqueta View que contendrá al Icono y a la caja de texto, poa si decirlo, este es un componente agrupado con otros componentes para ser uno solo. Sus propiedades ya están predefinidas a excepcion de las propiedades/parametros recibidas por la función. A continuación, este será el resultado de una caja de texto:



Código de componente: Label

```
const Label = ({
    size = 22,
    text = 'Letrero sin editar',
    weigth = '',
    align = 'left',
    color = 'black'
}) => {
  return (
    <View style = {styles.label}>
      <Text style = {{
        fontSize: size,
        fontWeight: weigth,
        textAlign: align,
        color: color
      }}>
        {text}
      </Text>
    </View>
  )
```

Ilustración 4. Código de componente: Label.js

Para este componente se declararon las siguientes propiedades:

size	Para definir el tamaño de la letra del componente.
text	El texto que se mostrara dentro del componente.
weigth	Se declara si este es del tipo “Normal, negrita, itálica, etc.”
align	Para donde estará alineado el texto.
color	El color que llevara el texto que se determine.

Este componente como tal, es para presentar texto, este es muy necesario, ya que se aplican los estilos por medio de los parámetros en la función sin necesidad de crear muchos Text y darle estilos a cada uno, repitiendo código, por lo tanto, es muy útil para colocar un texto, como títulos especialmente. Resultado:

Resultado de pantallas

Código y pantalla Inicio

El siguiente código este compuesto por la importación de los componentes de Label y Button, cada uno cambiando la configuración de cada propiedad editable.

Ilustración 5. Resultado final de implementación de componentes.

Finalmente, el componente de StackNavigator (Ilustración 1), será llamado a App.js para mostrar las pantallas anteriores (Ilustración 5).

```
import { StyleSheet, Text, View, TextInput, Animated } from 'react-native';
import  Login  from './src/screens/Login';
import Inicio from './src/screens/Inicio';
import StackNavigator from './src/StackNavigator';

const  App = () => {
  return (
    <StackNavigator />
  );
}

export default App;
```