# Анимация в React Native

### Цель работы

В этой работе мы рассмотрим основные принципы анимации, которые можно реализовать с помощью React Native.

### Задания для выполнения

1. Используя официальную документацию https://reactnative.dev/docs/animated ознакомьтесь с синтаксисом:

const FadeInView = (props) => {

const fadeAnim = useRef(new Animated.Value(0)).current // Initial value for opacity: 0

useEffect(() => {

Animated.timing(

fadeAnim,

{

toValue: 1,

duration: 10000,

}

).start();

}, [fadeAnim])

return (

<Animated.View // Special animatable View

style={{

...props.style,

opacity: fadeAnim, // Bind opacity to animated value

}}

>

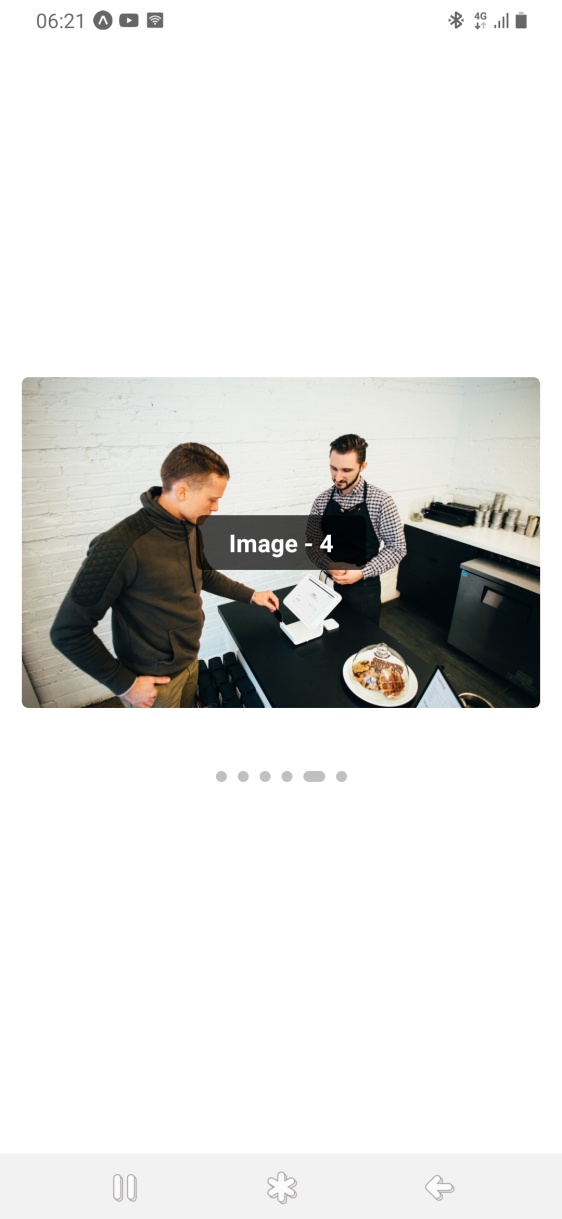
{props.children}

</Animated.View>

);

}

1. Ознакомьтесь с основными принципами анимации: <https://infogra.ru/ui/12-printsipov-primeneniya-animatsii-v-polzovatelskih-interfejsah>
2. Создайте анимацию для слайдера с различными картиками, добавьте другие компоненты на экран, используя созданные велкомскрины.



1. Создайте анимированный лаучскрин.

5. Поделитесь ссылкой на проект в Expo. Загрузить созданные приложение на GitHub в репозиторий Student, используя формат в названии Фамилия (латинскими буквами)\_5.

### Методические указания

Для создания слайдера можно воспользоваться кодом:

import React, { useRef } from "react";

import {

  SafeAreaView,

  ScrollView,

  Text,

  StyleSheet,

  View,

  ImageBackground,

  Animated,

  useWindowDimensions

} from "react-native";

const images = new Array(6).fill('https://images.unsplash.com/photo-1556740749-887f6717d7e4');

const App = () => {

  const scrollX = useRef(new Animated.Value(0)).current;

  const { width: windowWidth } = useWindowDimensions();

  return (

    <SafeAreaView style={styles.container}>

      <View style={styles.scrollContainer}>

        <ScrollView

          horizontal={true}

          style={styles.scrollViewStyle}

          pagingEnabled

          showsHorizontalScrollIndicator={false}

          onScroll={Animated.event([

            {

              nativeEvent: {

                contentOffset: {

                  x: scrollX

                }

              }

            }

          ])}

          scrollEventThrottle={1}

        >

          {images.map((image, imageIndex) => {

            return (

              <View

                style={{ width: windowWidth, height: 250 }}

                key={imageIndex}

              >

                <ImageBackground source={{ uri: image }} style={styles.card}>

                  <View style={styles.textContainer}>

                    <Text style={styles.infoText}>

                      {"Image - " + imageIndex}

                    </Text>

                  </View>

                </ImageBackground>

              </View>

            );

          })}

        </ScrollView>

        <View style={styles.indicatorContainer}>

          {images.map((image, imageIndex) => {

            const width = scrollX.interpolate({

              inputRange: [

                windowWidth \* (imageIndex - 1),

                windowWidth \* imageIndex,

                windowWidth \* (imageIndex + 1)

              ],

              outputRange: [8, 16, 8],

              extrapolate: "clamp"

            });

            return (

              <Animated.View

                key={imageIndex}

                style={[styles.normalDot, { width }]}

              />

            );

          })}

        </View>

      </View>

    </SafeAreaView>

  );

}

const styles = StyleSheet.create({

  container: {

    flex: 1,

    alignItems: "center",

    justifyContent: "center"

  },

  scrollContainer: {

    height: 300,

    alignItems: "center",

    justifyContent: "center"

  },

  card: {

    flex: 1,

    marginVertical: 4,

    marginHorizontal: 16,

    borderRadius: 5,

    overflow: "hidden",

    alignItems: "center",

    justifyContent: "center"

  },

  textContainer: {

    backgroundColor: "rgba(0,0,0, 0.7)",

    paddingHorizontal: 24,

    paddingVertical: 8,

    borderRadius: 5

  },

  infoText: {

    color: "white",

    fontSize: 16,

    fontWeight: "bold"

  },

  normalDot: {

    height: 8,

    width: 8,

    borderRadius: 4,

    backgroundColor: "silver",

    marginHorizontal: 4

  },

  indicatorContainer: {

    flexDirection: "row",

    alignItems: "center",

    justifyContent: "center"

  }

});

export default App;

### Контрольные вопросы

1. Что такое параллакс-эффект?

### Дополнительные задания

## Добавьте кнопку на экран и создайте ему анимацию.