

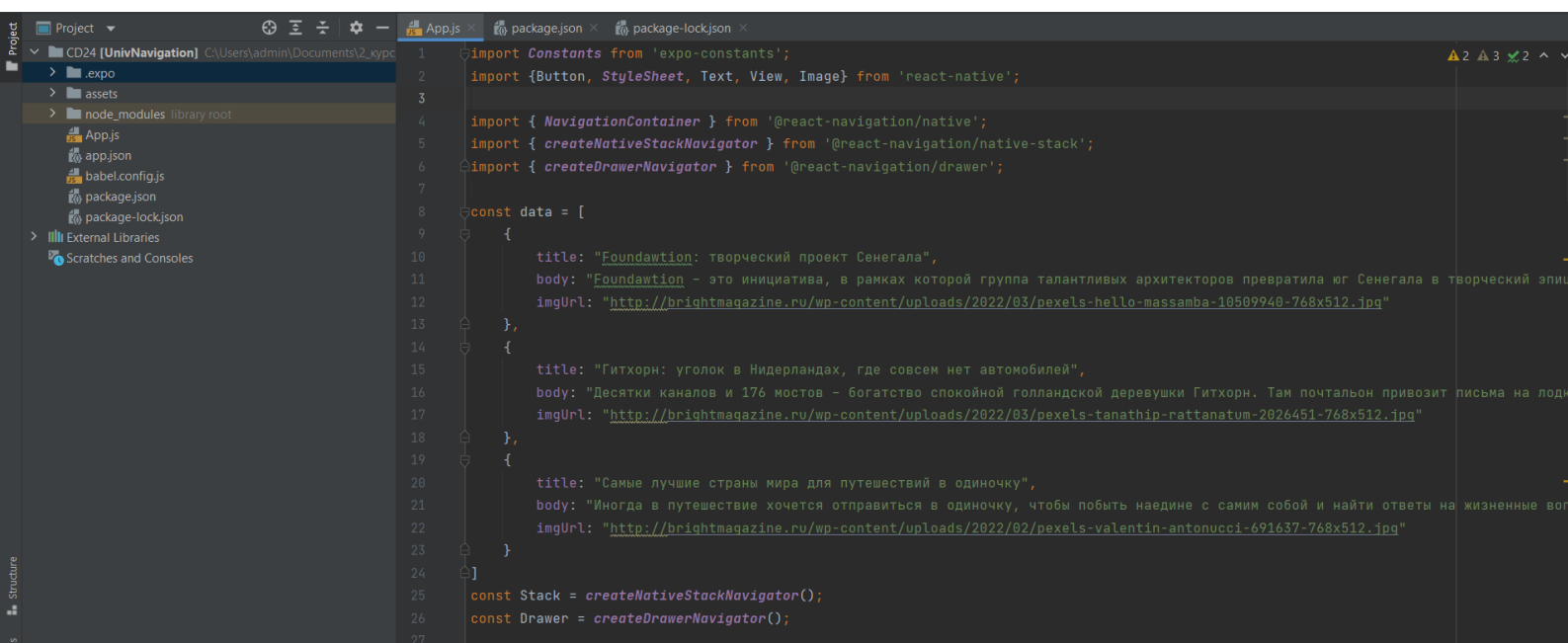
# Отчёт по проделанной работе на семинаре по мобильной разработке (15.04.2022).

(выполнил Валяев Георгий Анатольевич)

React-native – один из интересных и богатых фреймворков, наследующих большинство свойств фреймворка React (в рамках разработки нативных веб-приложений), разработанный командой Facebook для создания и написания приложений под мобильные устройства с поддержкой ОС Android, iOS и других систем.

В этой методичке мы рассмотрим процесс создания приложения с элементами стандартной навигации (например, «навигация по стеку»). Вдобавок к этому мы дополнительно изучим разработку навигации по боковой панели через Navigation Drawer. Таким образом, реализуя мини-журнал с сайта Bright, будут разобраны принципы создания способов внутренней навигации в приложении несколькими способами.

В одном главном файле App.js мы будем создавать все необходимые нам компоненты совместно с навигацией, так что рассмотрим его поподробнее:



```
1 import Constants from 'expo-constants';
2 import {Button, StyleSheet, Text, View, Image} from 'react-native';
3
4 import { NavigationContainer } from '@react-navigation/native';
5 import { createNativeStackNavigator } from '@react-navigation/native-stack';
6 import { createDrawerNavigator } from '@react-navigation/drawer';
7
8 const data = [
9   {
10     title: "Foundawtion: творческий проект Сенегала",
11     body: "Foundawtion - это инициатива, в рамках которой группа талантливых архитекторов превратила юг Сенегала в творческий эпицентр",
12     imgUrl: "http://brightmagazine.ru/wp-content/uploads/2022/03/pexels-hello-massamba-10509940-768x512.jpg"
13   },
14   {
15     title: "Гитхорн: уголок в Нидерландах, где совсем нет автомобилей",
16     body: "Десятки каналов и 176 мостов - богатство спокойной голландской деревушки Гитхорн. Там почтальон привозит письма на лодке",
17     imgUrl: "http://brightmagazine.ru/wp-content/uploads/2022/03/pexels-tanathip-rattanatum-2026451-768x512.jpg"
18   },
19   {
20     title: "Самые лучшие страны мира для путешествий в одиночку",
21     body: "Иногда в путешествие хочется отправиться в одиночку, чтобы побыть наедине с самим собой и найти ответы на жизненные вопросы",
22     imgUrl: "http://brightmagazine.ru/wp-content/uploads/2022/02/pexels-valentin-antonucci-691637-768x512.jpg"
23   }
24 ]
25
26 const Stack = createNativeStackNavigator();
27 const Drawer = createDrawerNavigator();
```

```

App.js x package.json x package-lock.json x
25 const Stack = createNativeStackNavigator();
26 const Drawer = createDrawerNavigator();
27
28 // Главный экран
29 const MainScreen = ({ navigation, route }) => {
30   return (
31     <View style={styles.container}>
32       <Text style={styles.header}>{data[0].title}</Text>
33       <Image source={{ uri: data[0].imgUrl }} style={styles.logoStyle}/>
34       <Text style={styles.body}>{data[0].body}</Text>
35       <View style={styles.button_view}>
36         <Button title="Следующая страница"
37           onPress={() => navigation.push('Вторая страница', { screen: "Второй" })} color="#000"/>
38       </View>
39       <Text style={styles.bold_text}>{route.params.screen} экран</Text>
40     </View>
41   );
42 };
43
44 // Второй экран
45 const SecondScreen = ({ navigation, route }) => {
46   return (
47     <View style={styles.container}>
48       <Text style={styles.header}>{data[1].title}</Text>
49       <Image source={{ uri: data[1].imgUrl }} style={styles.logoStyle}/>
50       <Text style={styles.body}>{data[1].body}</Text>
51       <View style={styles.button_view}>
52         <Button title="Последняя страница"
53           onPress={() => navigation.push('Последняя страница', { screen: "Третий" })} color="#000"/>
54     </View>

```

```

App.js x package.json x package-lock.json x
52       <Button title="Последняя страница"
53         onPress={() => navigation.push('Последняя страница', { screen: "Третий" })} color="#000"/>
54     </View>
55     <View style={styles.button_view_2}>
56       <Button title="Назад" onPress={() => navigation.goBack()} color="#000"/>
57     </View>
58   </View>
59 );
60 };
61
62 // Третий экран
63 const ThirdScreen = ({ navigation, route }) => {
64   return (
65     <View style={styles.container}>
66       <Text style={styles.header}>{data[2].title}</Text>
67       <Image source={{ uri: data[2].imgUrl }} style={styles.logoStyle}/>
68       <Text style={styles.body}>{data[2].body}</Text>
69       <View style={styles.button_view}>
70         <Button title="Перейти на первую страницу" onPress={() =>
71           navigation.push('Первая страница', { screen: "Главный" })} color="#000"/>
72       </View>
73       <View style={styles.button_view}>
74         <Button title="Назад" onPress={() => navigation.goBack()} color="#000"/>
75       </View>
76       <View style={styles.button_view}>
77         <Button title="На главную страницу" onPress={() => navigation.popToTop()} color="#000"/>
78       </View>
79       <Text style={styles.bold_text}>{route.params.screen} экран</Text>
80     </View>
81   );

```

```

79         <Text style={styles.bold_text}><route.params.screen> экран</Text>
80     </View>
81 };
82
83
84 // Указываем везде initialParams для доступа к route.params.screen
85 // (из-за того, что Drawer.Navigator просто возвращает экран, а передача доп параметра screen происходит в кнопке )
86 function DrawerNavigation() {
87     return (
88         <Drawer.Navigator>
89             <Drawer.Screen name="Первая страница" component={MainScreen} initialParams={{ screen: "Главный"}}/>
90             <Drawer.Screen name="Вторая страница" component={SecondScreen} initialParams={{ screen: "Второй"}}/>
91             <Drawer.Screen name="Последняя страница" component={ThirdScreen} initialParams={{ screen: "Третий"}}/>
92         </Drawer.Navigator>
93     )
94 }
95
96 export default function App() {
97     return (
98         <NavigationContainer>
99             <Stack.Navigator>
100                 {/* Первым подключаем Drawer Navigator как вложенный в Stack Navigator */}
101                 <Stack.Screen name="DrawerNavigator" component={DrawerNavigation} options={{ headerShown: false }}/>
102                 <Stack.Screen name="Первая страница" component={MainScreen} />
103                 <Stack.Screen name="Вторая страница" component={SecondScreen} />
104                 <Stack.Screen name="Последняя страница" component={ThirdScreen} />
105             </Stack.Navigator>
106         </NavigationContainer>
107     );
108 }

```

```

106     </NavigationContainer>
107 );
108 }
109
110 const styles = StyleSheet.create({
111     container: {
112         paddingTop: Constants.statusBarHeight,
113         flex: 1,
114         backgroundColor: '#fff',
115         alignItems: 'center',
116         justifyContent: 'center',
117     },
118     button_view: {
119         margin: 15,
120     },
121     button_view_2: {
122         margin: 10
123     },
124     bold_text: {
125         margin: 10,
126         fontSize: 18,
127         fontWeight: "bold",
128     },
129     logoStyle: {
130         width: 350,
131         height: 220,
132         marginLeft: 20,
133         borderRadius: 15
134     },

```

```
App.js  package.json  package-lock.json
124     bold_text: {
125       margin: 10,
126       fontSize: 18,
127       fontWeight: "bold",
128     },
129     logoStyle: {
130       width: 350,
131       height: 220,
132       marginLeft: 20,
133       borderRadius: 15
134     },
135     header: {
136       color: 'black',
137       fontSize: 28,
138       fontWeight: "bold",
139       marginBottom: 20,
140       marginLeft: 20,
141       marginTop: 10,
142     },
143     body: {
144       color: 'black',
145       fontSize: 18,
146       marginLeft: 20,
147       marginTop: 25,
148       marginRight: 40
149     }
150   });
```

Начнём по порядку. В верхней части файла мы импортируем все необходимые нам модули для настройки навигации (подробнее остановимся на них позже) и рендеринга всех объектов на экран приложения.

Затем (из методички по главным компонентам) мы берём всё необходимое наполнение для страничек мини-журнала и оборачиваем в отдельные структуры в едином и целом объекте `data`.

После мы инициализируем новые объекты навигации, классы и модули которых они импортировали ранее (`createNativeStackNavigator()` из `@react-navigation/native-stack`, а также `createDrawerNavigator()` из `@react-navigation/drawer`), первый из которых мы будем использовать для создания конкретного стека экранов: первый, из которых мы увидим, будет с самого конца стека, согласно его терминологии, и так далее до конца, а потом по обратной стороне аналогичная история. Касательно второго объекта стоит сказать, что он как раз-таки будет служить вложенным в стек объектом, на который мы можем перейти на устройстве, совершив свайп пальцем вправо и

выбрав уже любой экран. Тем самым получится универсальная навигация по приложению среди экранов при помощи Navigation Drawer.

Далее происходит процесс построения каждого экрана по отдельности, у каждого из которых своё наполнение и алгоритм наполнения объектами и данными:

Единственное, что хочется подметить общим между ними – это структура переменной со стрелочной функцией, внутри которой отправляется переменная («navigation»), отслеживающая состояние в процессе навигации (отслеживание текущего экрана среди остальных), и переменная, в которую будут отправляться данные для отображения или другого функционала («route»). Вдобавок к этому в каждом экране прописана структура отображения данных из ранее созданного объекта data:

...

<Text> текст заголовка </Text>

<Image source={ссылка на данные внутри объекта data}... />

<Text> текст наполнения статьи </Text>

...

- 1) Главный экран. В большом объекте View мы прописываем отображение данных, после чего в отдельном объекте View создаём кнопку <Button> для дальнейшей навигации к следующему экрану с заголовком и обработчиком нажатия `onPress={() => navigation.push('Вторая страница', { screen: "Вторая" })}`, который переводит вниз по стеку экранов навигации при реагировании.
- 2) Второй экран. Картина немного похожая, но только теперь мы добавляем к стандартному объекту <Button> на следующий экран (`onPress={() => navigation.push('Последняя страница', { screen: "Третья" })}`) дополнительно в другом объекте View ещё одну кнопку «Назад»,

которая отвечает при реагировании перемещение вверх по стеку экранов: `onPress={() => navigation.goBack()}`.

- 3) Тут уже поинтереснее картина складывается. Помимо наполнения данными последнего экрана приложения, здесь мы размещаем уже 3 кнопки, у 2 из которых идентичное предназначение: перевод на главный экран. Начнём с первой кнопки в своём объекте View, которая переводит на главный экран при помощи обработчика нажатия на кнопку `onPress={() => navigation.push('Первая страница', { screen: "Главная" })}`, а третья по порядку кнопка, но похожая по функционалу реализована через свой обработчик `onPress={() => navigation.popToTop()}` в своём объекте View. Основное отличие между ними заключается в том, что в первом случае мы ссылаемся на первый экран при помощи перевода по стеку вверх по своего рода идентификатору (`screen: "Главная"`), а уже у второго способа идёт перемещение на самый верх стека экранов через метод `.popToTop()` у `navigation`. А последняя кнопка между ними копирует функционал кнопки «Назад» у второго экрана приложения.

После основного блока мы заканчиваем рендеринг всех компонентов приложения при помощи ранее созданных 2 объектов навигации, один из которых входит в другой. В функции `DrawerNavigation()` создания боковой панели навигации мы внутри объекта `Drawer.Navigator` располагаем наши экраны с названиями, назначением компонентов отображения и инициализирующими параметрами для перемещения между экранами приложения (`initialParams={{ screen: ... }}`).

Далее в главной функции компонента App, которую мы также экспортируем в рендеринг проекта, в основном объекте `NavigationContainer`, который мы в начале импортировали из модуля `@react-navigation/native`, располагается весь стек экранов совместно с ранее созданной боковой панелью (с дополнительным параметром `options={{ headerShown: false }}`) для доступа

пользователю на переднем плане параллельно с первым экраном. Остальные экраны (`Stack.Screen`) в объекте стека `Stack.Navigator` объявляются с названием и предназначенным по функционалу компонентом, созданном ранее в файле приложения.

Под конец мы объявляем все стили, которые были и будут применены ко всем компонентам и объектам, с вполне понятными и базовыми назначениями: внешний отступ от остальных объектов экрана (`margin`), цвет внешнего фона (`backgroundColor`), расположение по центру элементов (`alignItems & justifyContent`), цвет, размер и стиль шрифта, ширина и высота элемента.

Полный анализ всего проекта успешно завершён. Смотрим на получившийся результат!

15:22 @

4%

← Первая страница

## Foundawtion: творческий проект Сенегала



Foundawtion – это инициатива, в рамках которой группа талантливых архитекторов превратила юг Сенегала в творческий эпицентр.

СЛЕДУЮЩАЯ СТРАНИЦА

Главный экран



15:22 @

4%

Первая страница

Вторая страница

Последняя страница

еский



, в  
ЛИВЫХ

нтр.





## Гитхорн: уголок в Нидерландах, где совсем нет автомобилей



Десятки каналов и 176 мостов – богатство спокойной голландской деревушки Гитхорн. Там почтальон привозит письма на лодке, а автомобилей нет и в помине. Идеальное место, чтобы отдохнуть от городского шума и суеты.

[ПОСЛЕДНЯЯ СТРАНИЦА](#)[НАЗАД](#)

## Самые лучшие страны мира для путешествий в одиночку



Иногда в путешествие хочется отправиться в одиночку, чтобы побыть наедине с самим собой и найти ответы на жизненные вопросы. Сегодня мы расскажем вам о самых безопасных странах мира именно для таких путешествий.

[ПЕРЕЙТИ НА ПЕРВУЮ СТРАНИЦУ](#)[НАЗАД](#)[НА ГЛАВНУЮ СТРАНИЦУ](#)

15:22

4G+ 4%

Первая страница

Вторая страница

Последняя страница

раны  
твий в



ы  
й

кем  
нах  
ествий.

ицу



Работа успешно закончена!

### Контрольные вопросы

1. Как установить зависимости для управляемого проекта Ехро?
1. Этот процесс можно осуществить пропиской в терминале команды «ехро install (название зависимости/пакета)» или «npm install (название зависимости/пакета)».