|  | Министерство образования и науки Российской Федерации  Федеральное государственное автономное образовательное учреждение высшего образования «Уральский федеральный университет имени первого Президента России Б.Н.Ельцина» (УрФУ)  ИРИТ-РТФ  Базовая кафедра «Аналитика больших данных и методы видео анализа» |
| --- | --- |

Report on the project

**«Свадебное приложение - ParaPlan»**

Student: Сушков Дмитрий Андреевич РИМ - 130980

Student: Савченко Сергей Дмитриевич РИМ - 130980

Yekaterinburg

2024

**Оглавление**

[**Введение 3**](#_xv2e1d1avm79)

[**Постановка задачи 4**](#_u7nxwpl2l1ba)

[**Архитектура проекта 6**](#_ivip7kimvykz)

[**Описание реализации проекта 9**](#_wlp3vvnjucph)

[1. Файл App.js 9](#_y3pbm9v0xv86)

[2. Папка Assets 9](#_8kb6fth05i1v)

[3. Папка Components 10](#_ej5h7li1rjjf)

[4. Файл AppNavigator.js 11](#_yq8hge4ft6jf)

[5. Файл LoginScreen.js 11](#_vg68ndnewvah)

[6. Файл RegisterScreen.js 15](#_k8ikd17usit9)

[7. Файл database.js 18](#_msq9xqdafcl4)

[8. Файл CatalogScreen.js 19](#_qeu4kr9g12i5)

[9. Файл ConfScreen.js 21](#_6ug3ugxcyku9)

[10. Файл MainScreen.js 22](#_pohu8cuh7lhy)

[11. Файл PageIndicator.js 24](#_63ba43sh0m4p)

[12. Файл Onboard1.js, Onboard2.js, Onboard3.js, Onboard4.js, Onboard5.js 25](#_lo0hr2qc93b0)

[**Итоги проекта 29**](#_18xrz6grd4fq)

# **Введение**

**Название проекта:** «ПараПлан»

**Актуальность проекта:**

В настоящее время, свадебная индустрия продолжает процветать, и многие люди стремятся сделать свои свадьбы особенными и запоминающимися. Кроме того, актуальность темы становится очевидной в контексте постоянного развития цифровых технологий и онлайн-сервисов. Пары, планирующие свадьбу, все чаще обращаются к онлайн-инструментам для поиска и выбора места проведения церемонии, подбора услуг и декораций, а также для взаимодействия с поставщиками свадебных услуг. В этом контексте конфигуратор свадебных мероприятий представляет собой современный и удобный инструмент, который отвечает потребностям цифровой эпохи и обеспечивает персонализированный подход к планированию свадебных торжеств.

**Цель проекта:**

Основной целью проекта является разработка мобильного приложения конфигуратора свадебных мероприятий(«ПараПлан») для платформы Android и iOS.

# **Постановка задачи**

**Задачи проекта:**

1. Проектирование проекта:
   * Определение цели и задач
   * Определение основного функционала
   * Разработка концептуального проекта
2. Техническая реализация:
   * Реализация стартового экрана
   * Реализация Входа и Регистрации
   * Реализация БД, где хранятся данные входа и регистрации
   * Реализация системы навигации с использованием React Navigation, включая нижнее меню для переключения между основными экранами.
   * Реализация экрана с каталогом и предлагаемой информацией
   * Реализация экрана конфигуратора с возможностью сохранения/добавления/удаления
3. Тестирование продукта:
   * Провести проверку всех ключевых функций приложения с целью обнаружения и устранения любых ошибок.
   * Улучшить работу приложения, чтобы оно работало более эффективно, и обеспечить его надежную и стабильную функциональность.

**Бизнес-требования:**

1. Мобильное приложение должно иметь простой и интуитивный интерфейс для всех типов пользователей
2. Мобильное приложение должно предоставлять пользователю все стандартные функции
3. Мобильное приложение должно предоставлять пользователю настройку создания собственного конфигуратора
4. Мобильное приложение должно обеспечивать безопасную и удобную аутентификацию пользователей

**Риски проекта:**

Основной риск проекта заключается в нехватке времени для реализации всех функций проекта, а также в уходе из РФ некоторых сервисов, используемых в разработке сайта и приложения. Проект должен выйти полностью завершённым, что бы первые впечатление пользователей сформировалось положительным, за время отладки проекта возможно появление более крупного конкурента.

**Функциональные-требования:**

1. Мобильное приложение должно иметь регистрацию и вход
2. Мобильное приложение должно базу данных для хранения логина и пароля
3. Мобильное приложение должно предоставлять корректную информацию об услугах
4. Мобильное приложение должно иметь точный конфигуратор, который хранит сведения добавленных услуг
5. Мобильное приложение должно иметь удобную навигацию и правильную связь

# 

# **Архитектура проекта**

Была составлена схема проекта “Компоненты и интерфейс” (рис.1)

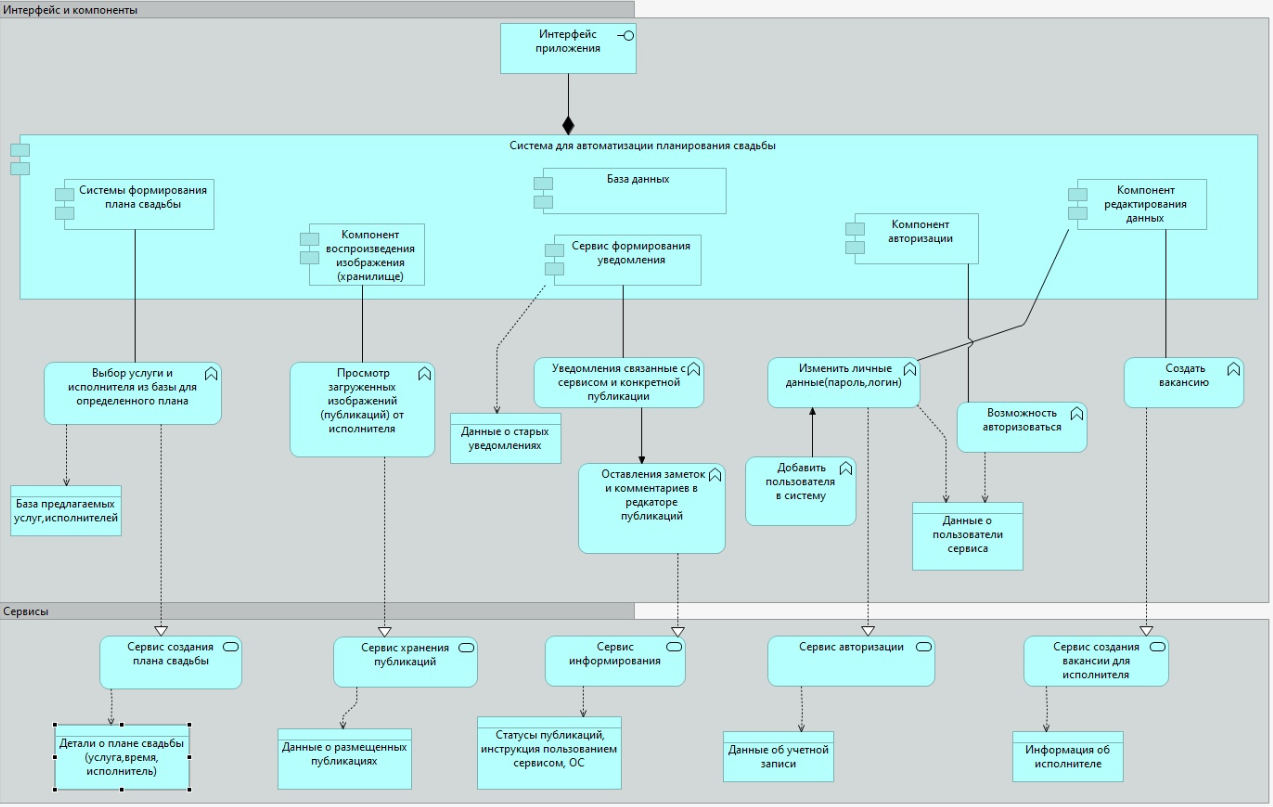


Рисунок 1 - “Компоненты и интерфейс”

Хотелось бы также отметить, что данная схема будет дорабатываться в будущем и на данный момент имеет уже POST-компоненты проекта.

Также была составлена схема “Работа сервиса в коммерческих условиях” (рис.2). Данная схема дает понимание коммуникации в сервисе, если бы данный продукт был введен или выкуплен коммерческим предприятием.

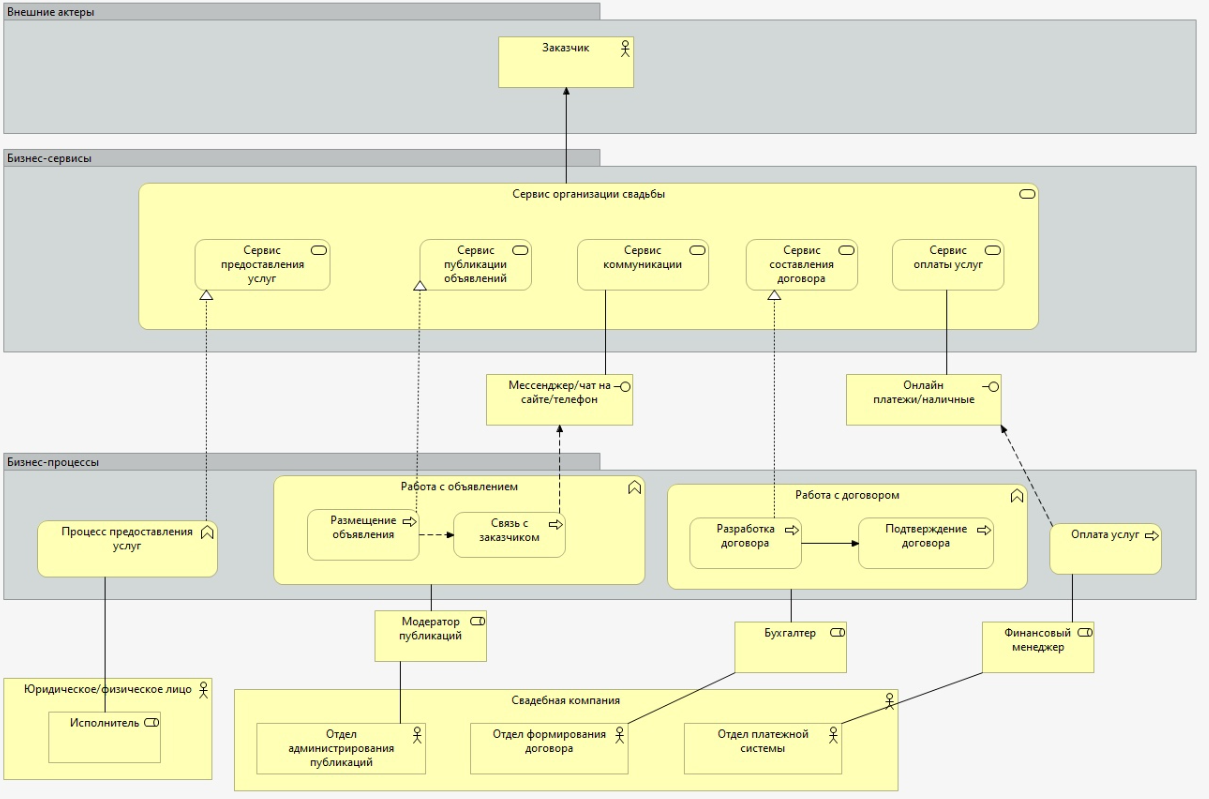


Рисунок 2 - “Работа сервиса в коммерческих условиях”

Также была продумана схема(рис.3), которая объединила в себе такие ключевые параметры, как:

1. Стейкхолдеры и вызовы
2. Проблематика
3. Цели
4. Функциональные требования

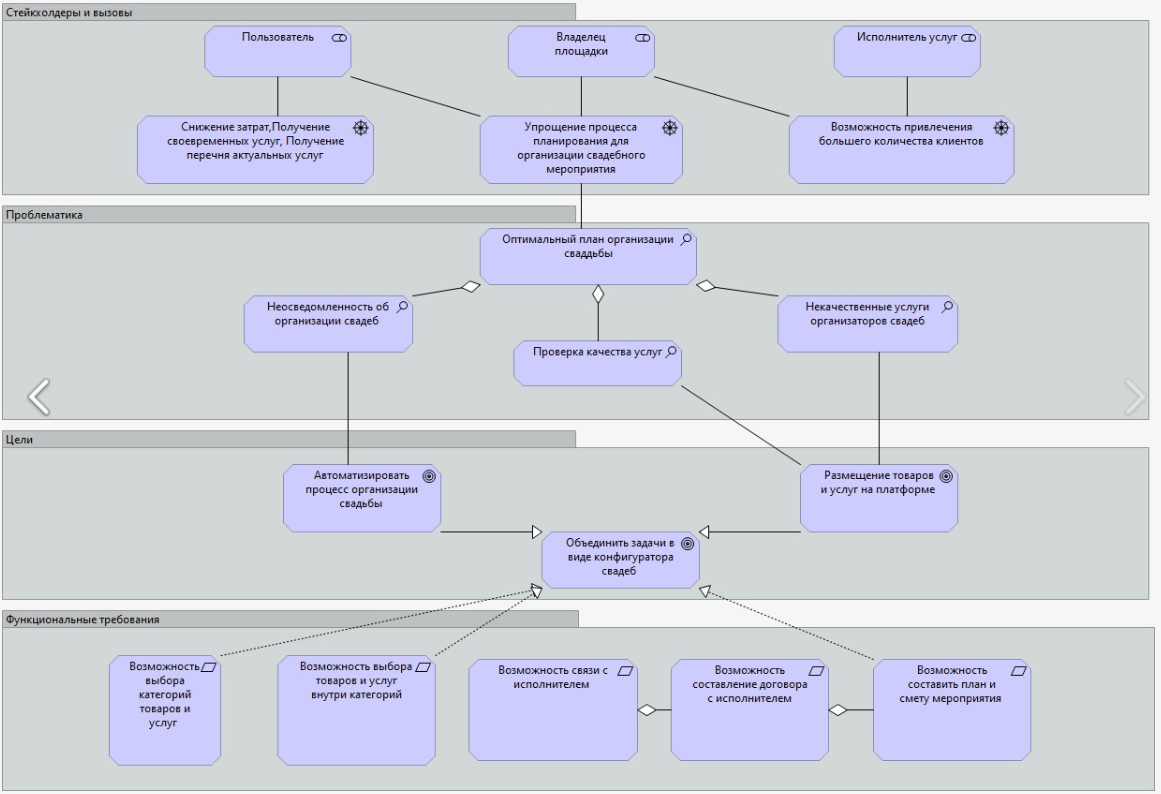


Рисунок 3 - Общая схема

**POST-задача**: На будущее была разработана архитектура (рис.5) переноса мобильного приложения в WEB-решение.

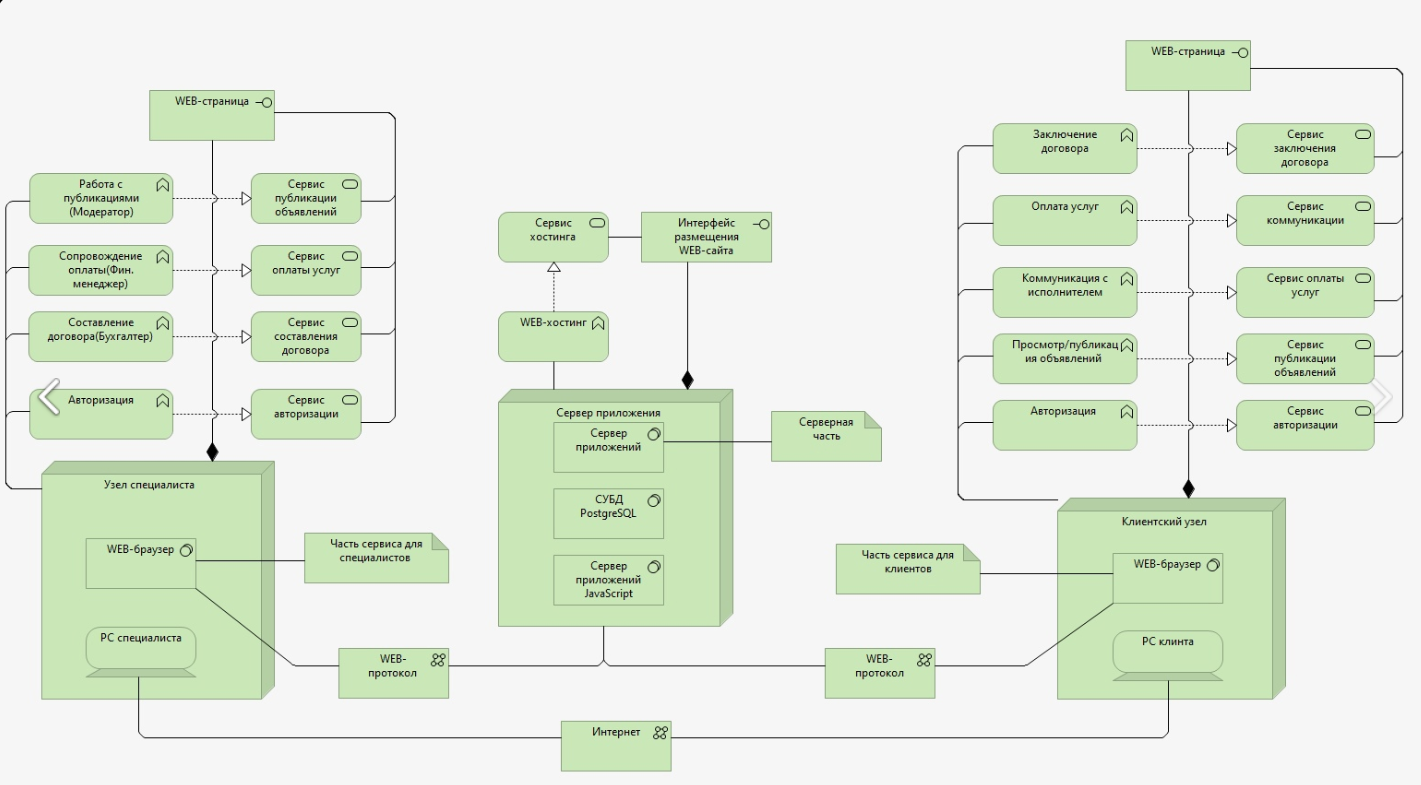


Рисунок 5

# **Описание реализации проекта**

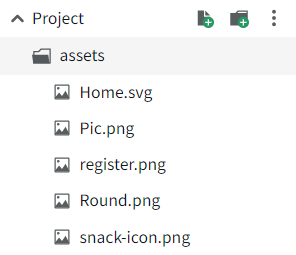
## **Файл App.js**

Данный фрагмент кода представляет собой файл, предназначенный для инициализации корневого компонента приложения на основе платформы React. В этом файле определяется компонент App, который использует хук useEffect для проверки состояния проведения вводного обучения и хук useState для управления этим состоянием. Также используется компонент NavigationContainer, предположительно из библиотеки React Navigation, для обеспечения навигации в приложении. Располагается код ресурса React Navigation внутри компонента AppNavigator.



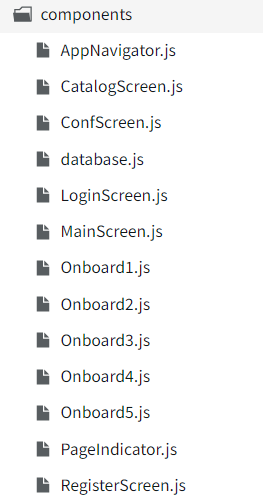
## **Папка Assets**

В папке мы храним изображения.



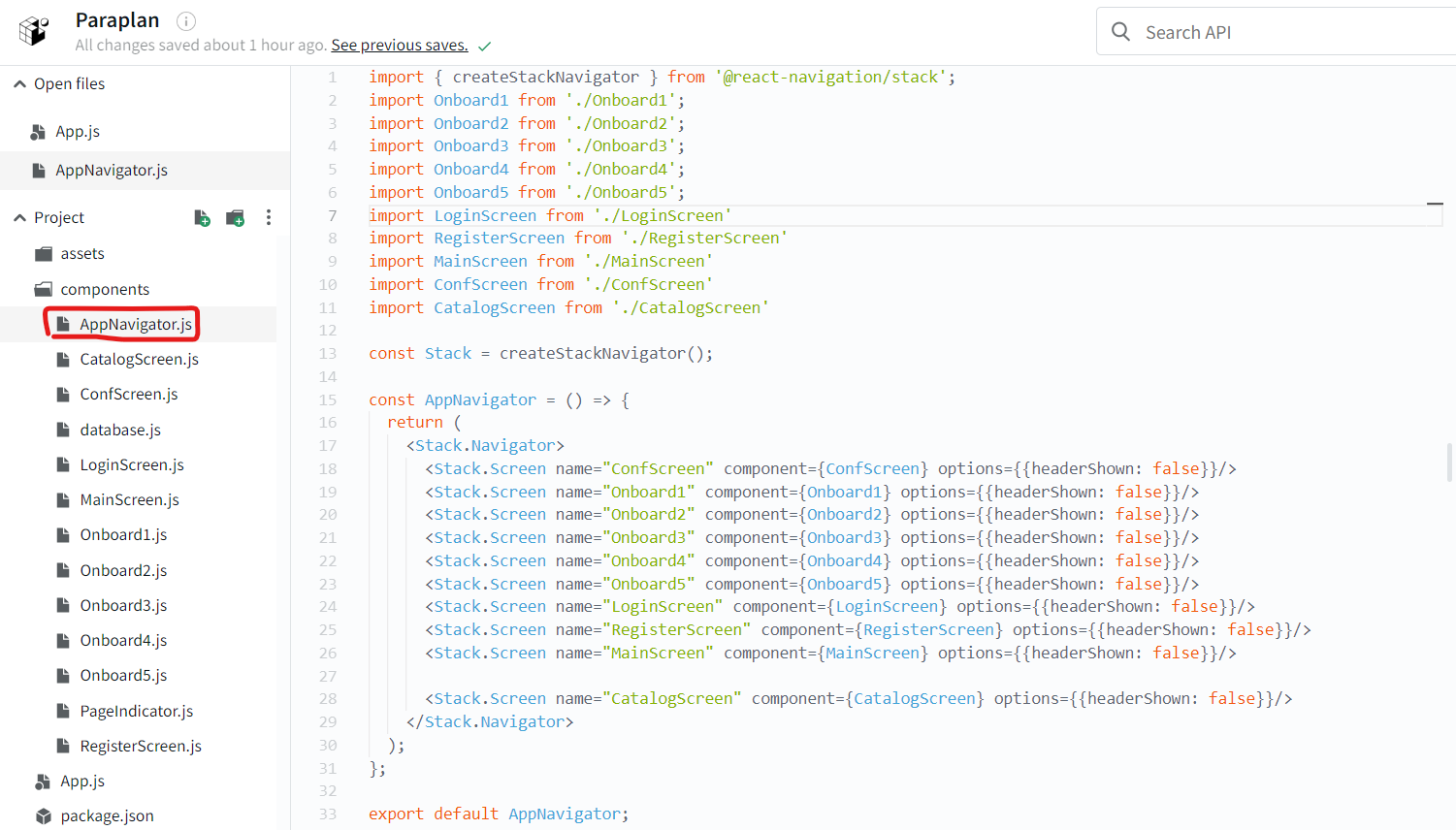
## **Папка Components**

В папке мы храним компоненты экранов и логики навигации.



## **Файл** **AppNavigator.js**

Данный код создает навигационный стек на основе React Navigation. В этом стеке определены различные экраны (Onboard1, Onboard2, Onboard3, Onboard4, Onboard5, LoginScreen, RegisterScreen, MainScreen, ConfScreen, и CatalogScreen) вместе с соответствующими компонентами для отображения каждого экрана. Компонент AppNavigator создает стек навигации, который включает определенные экраны. Каждый экран имеет свои собственные параметры опций, включая параметр headerShown: false, который указывает, что заголовок экрана не должен отображаться в каждом из определенных экранов. Работа с навигацией и отображение компонентов осуществляется с использованием компонента Stack.Navigator и Stack.Screen из библиотеки React Navigation.



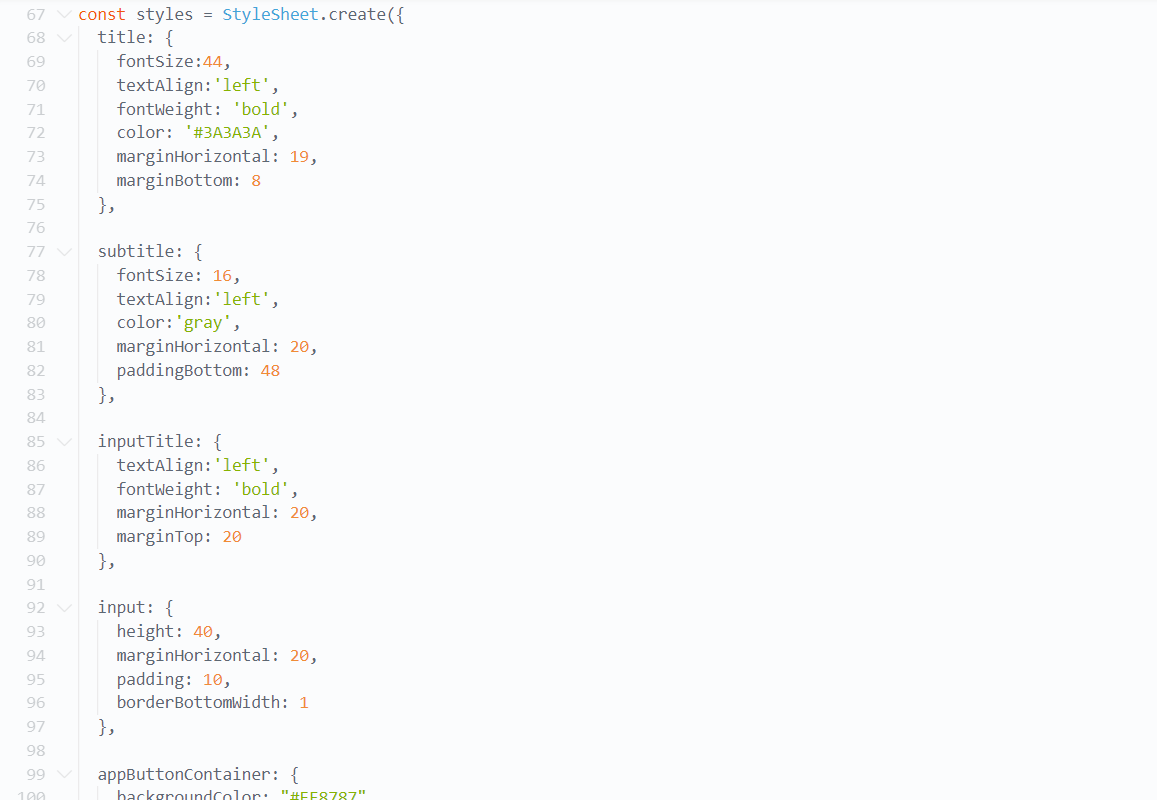
## **Файл** **LoginScreen.js**

Данный файл представляет компонент LoginScreen для мобильного приложения, который отвечает за экран входа пользователя. В файле определены следующие основные функции и компоненты:

* Создание компонента LoginScreen, который содержит состояния для электронной почты (email) и пароля (password) с помощью хука useState. Также использован хук useNavigation для управления навигацией.
* Определение функции handleLogin, которая вызывает функцию проверки учетных данных из базы данных (checkLoginCredentials) с передачей email и password. В зависимости от результата, происходит навигация либо на главный экран (MainScreen), либо выводится сообщение об ошибке через Alert.
* Создание компонента AppButton для отображения кнопки входа с заданными стилями.
* Рендеринг разметки экрана входа с использованием изображения, текстовых полей для ввода электронной почты и пароля кнопки для входа, а также ссылки для перехода на экран регистрации.
* Определение стилей для различных элементов интерфейса, таких как заголовок, подзаголовок, полей ввода, кнопок и текстов.









## **Файл** **RegisterScreen.js**

Данный файл представляет компонент RegisterScreen для мобильного приложения, отвечающий за экран регистрации нового пользователя. В файле определены следующие основные функции и компоненты:

* Создание компонента RegisterScreen, который содержит состояния для электронной почты (email) и пароля (password) с помощью хука useState. Также использован хук useNavigation для управления навигацией.
* Определение функции handleRegister, которая проверяет, что электронная почта и пароль заполнены, и затем вызывает функцию insertUser для регистрации нового пользователя. Если введенные данные неверны, выводится сообщение об ошибке через Alert.
* Создание компонента AppButton для отображения кнопки создания аккаунта с заданными стилями.
* Рендеринг разметки экрана регистрации с использованием текстовых полей для ввода электронной почты и пароля, кнопки для создания аккаунта, а также ссылки для перехода на экран входа.
* Определение стилей для различных элементов интерфейса, таких как заголовок, подзаголовок, полей ввода, кнопок и текстов.





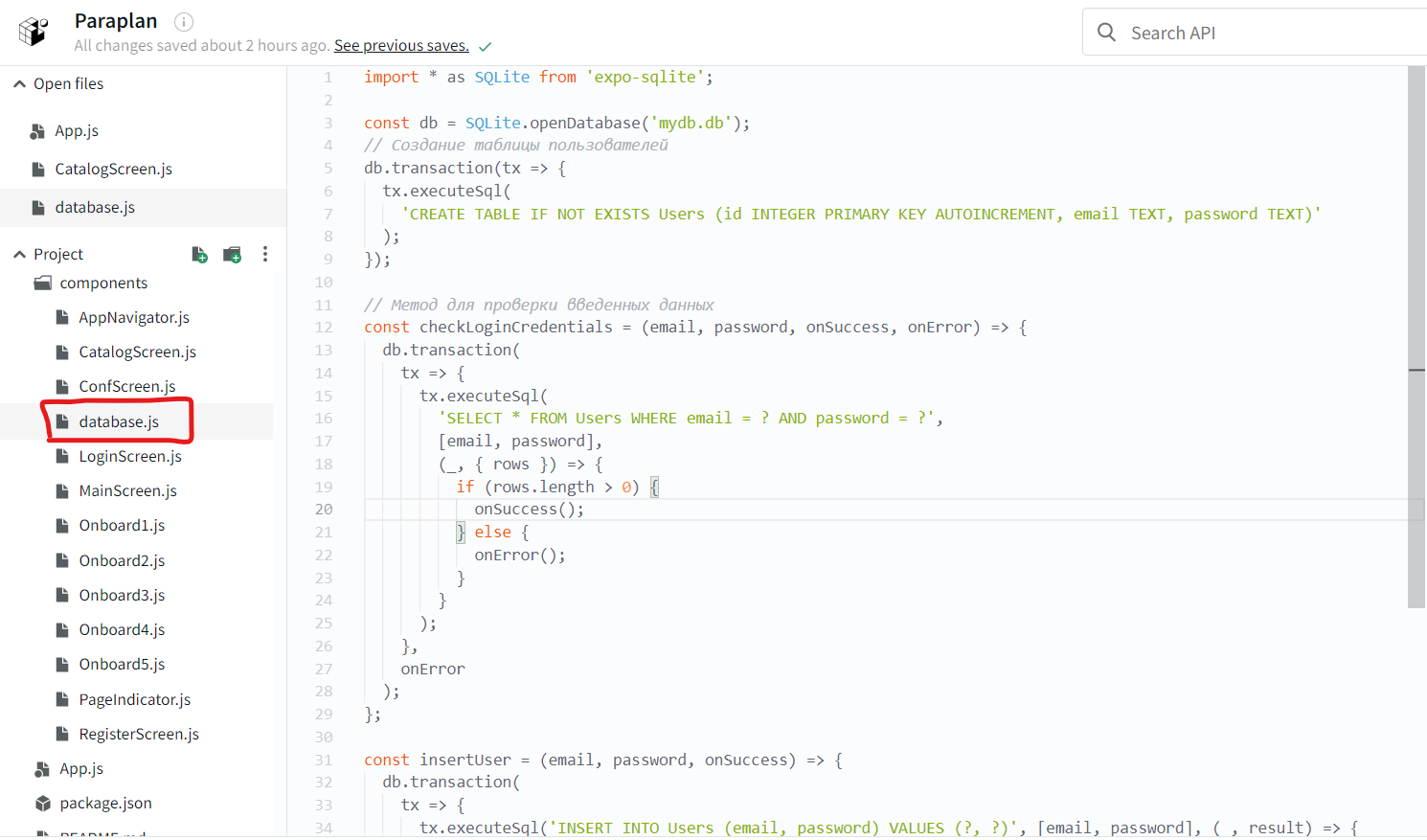




## **Файл** **database.js**

Предоставленный код на русском языке настраивает базу данных SQLite с использованием библиотеки expo-sqlite для приложения, разработанного в рамках фреймворка Expo. Он создает таблицу для пользователей и определяет методы для взаимодействия с этой базой данных.

* Создание таблицы: В коде используется оператор SQL CREATE TABLE IF NOT EXISTS, чтобы создать таблицу с названием Users в базе данных SQLite. В таблице содержатся столбцы id, который является автоинкрементируемым целочисленным первичным ключом, email и password. Это гарантирует создание таблицы, если она еще не существует.
* Метод для проверки учетных данных при входе в систему: Метод checkLoginCredentials принимает email и password в качестве входных параметров и определяет обратные вызовы для onSuccess и onError. Внутри этого метода выполняется SQL-запрос для проверки наличия строк в таблице Users, которые соответствуют предоставленному электронному адресу и паролю. Если совпадение найдено, вызывается обратный вызов onSuccess; в противном случае вызывается обратный вызов onError.
* Метод для вставки пользователя: Метод insertUser отвечает за вставку нового пользователя в таблицу Users. Он принимает email, password и обратный вызов onSuccess в качестве параметров. Внутри метода выполняется SQL-запрос для вставки предоставленного email и password в таблицу Users. Если вставка прошла успешно (т.е. если result.rowsAffected больше 0), вызывается обратный вызов onSuccess. Экспорт операторов: В конце файла методы insertUser и checkLoginCredentials экспортируются для обеспечения доступности их использования в других частях приложения.



## **Файл** **CatalogScreen.js**

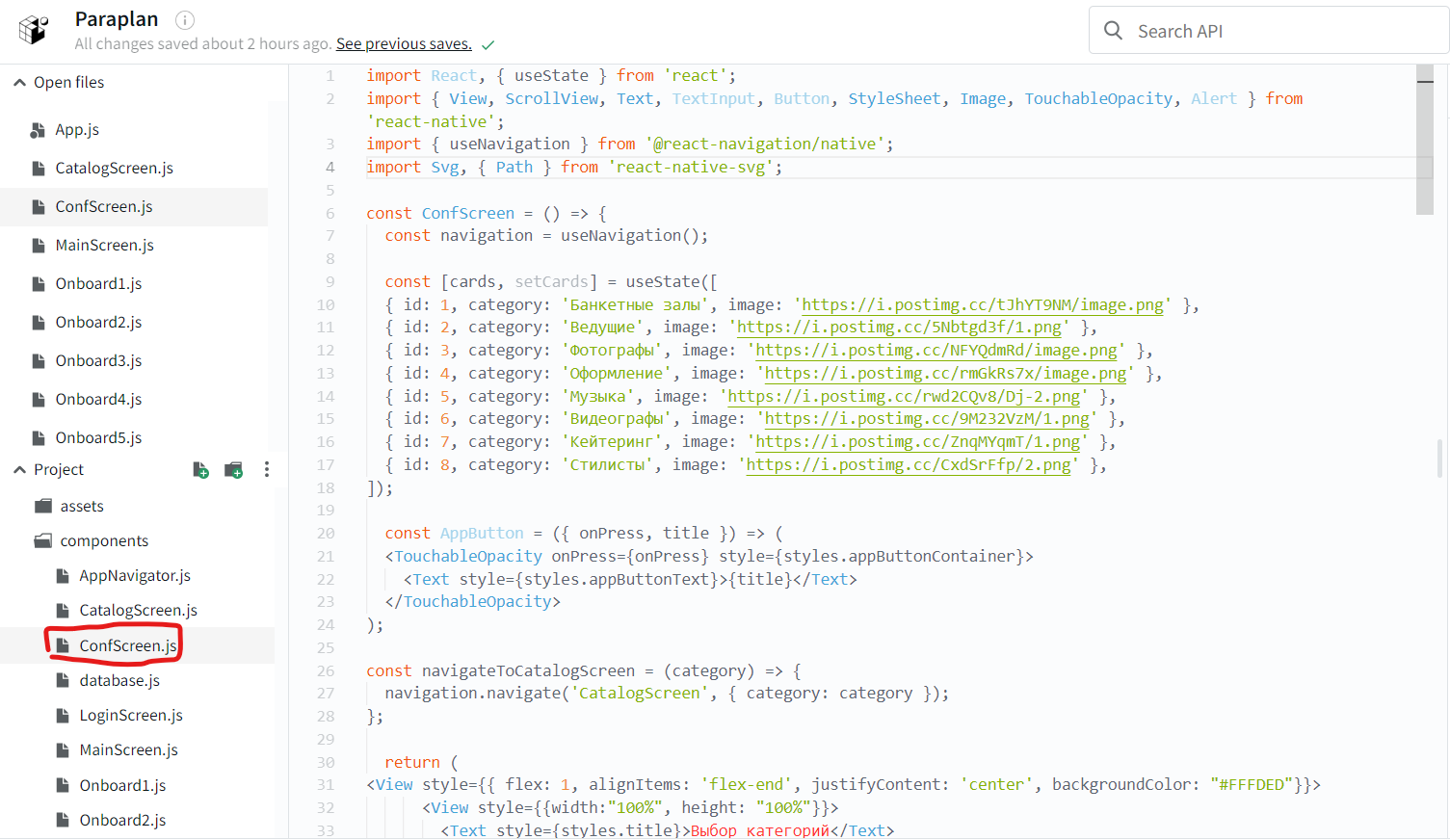
Содержит логику и информацию вкладки “Каталог”  






## **Файл** **ConfScreen.js**

Содержит логику и информацию вкладки “Конфигуратор”







## **Файл** **MainScreen.js**

Основные задачи компонента:

* Инициализирует переменные состояния, такие как 'email', 'password', 'weddingDate' и 'isDatePickerVisible', с использованием хука useState для обработки различных типов данных и видимости компонентов.
* Определяет функции showDatePicker, hideDatePicker и handleConfirm для управления видимостью и поведением компонента выбора даты.
* Определяет компонент AppButton, который инкапсулирует TouchableOpacity из React Native для создания настраиваемой стилизованной кнопки.







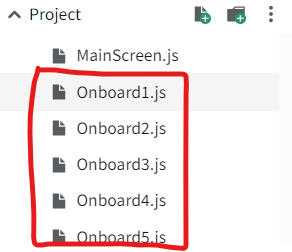
## **Файл** **PageIndicator.js**

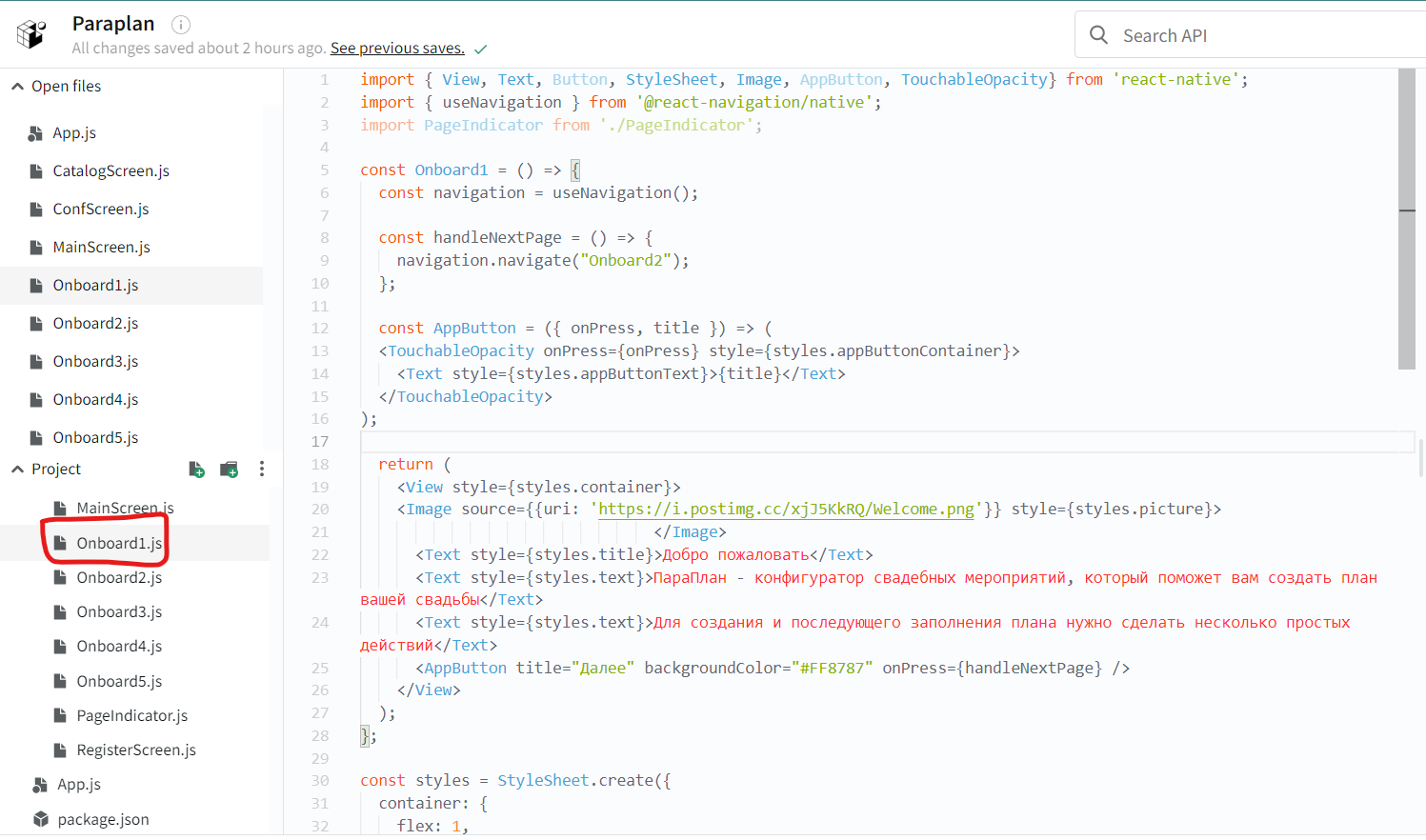
Этот файл определяет React Native-компонент под названием PageIndicator, который отображает серию индикаторов, обычно используемых для представления текущей страницы в многостраничном интерфейсе.

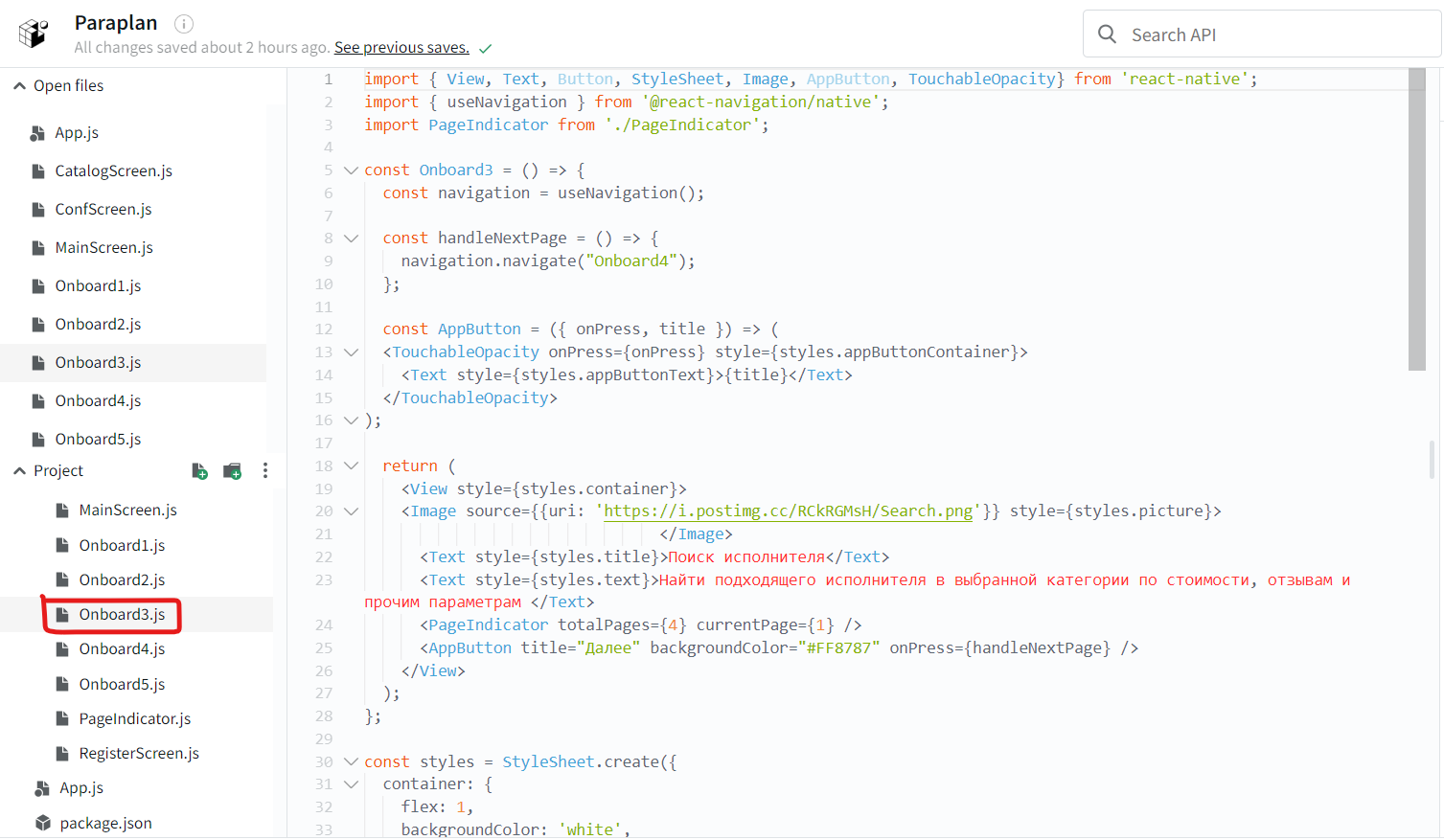
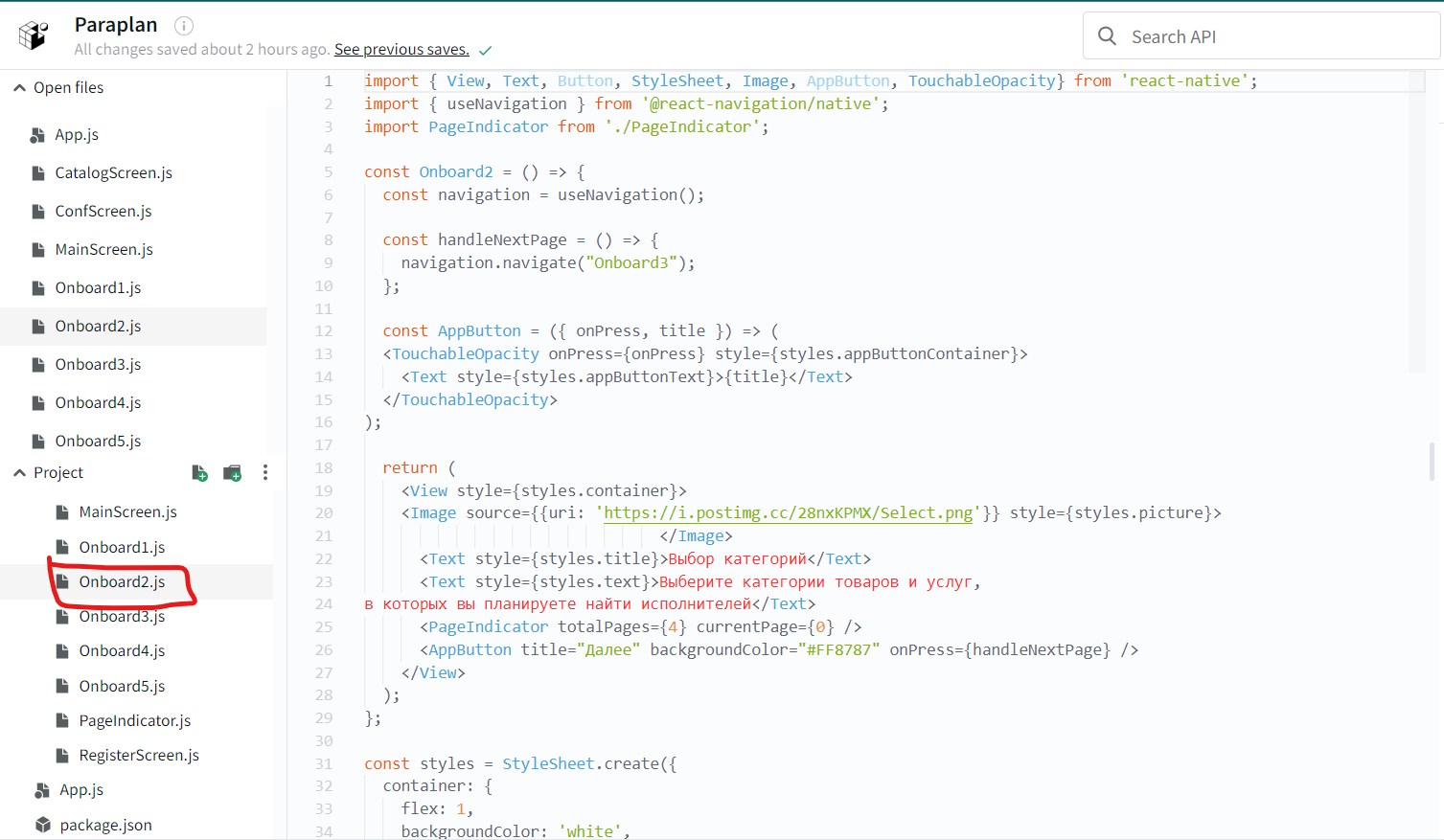


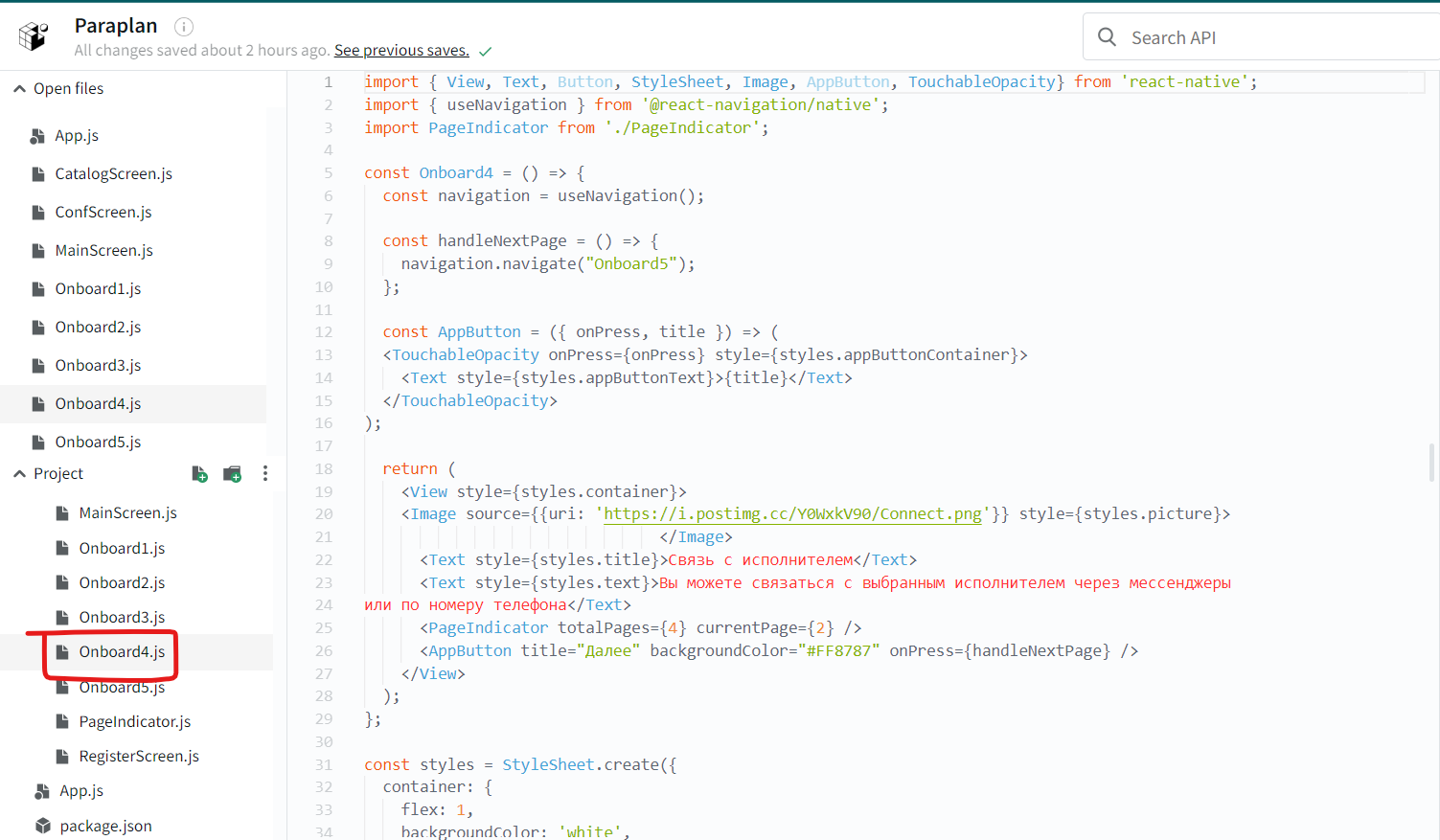
## **Файл** **Onboard1.js, Onboard2.js, Onboard3.js, Onboard4.js, Onboard5.js**

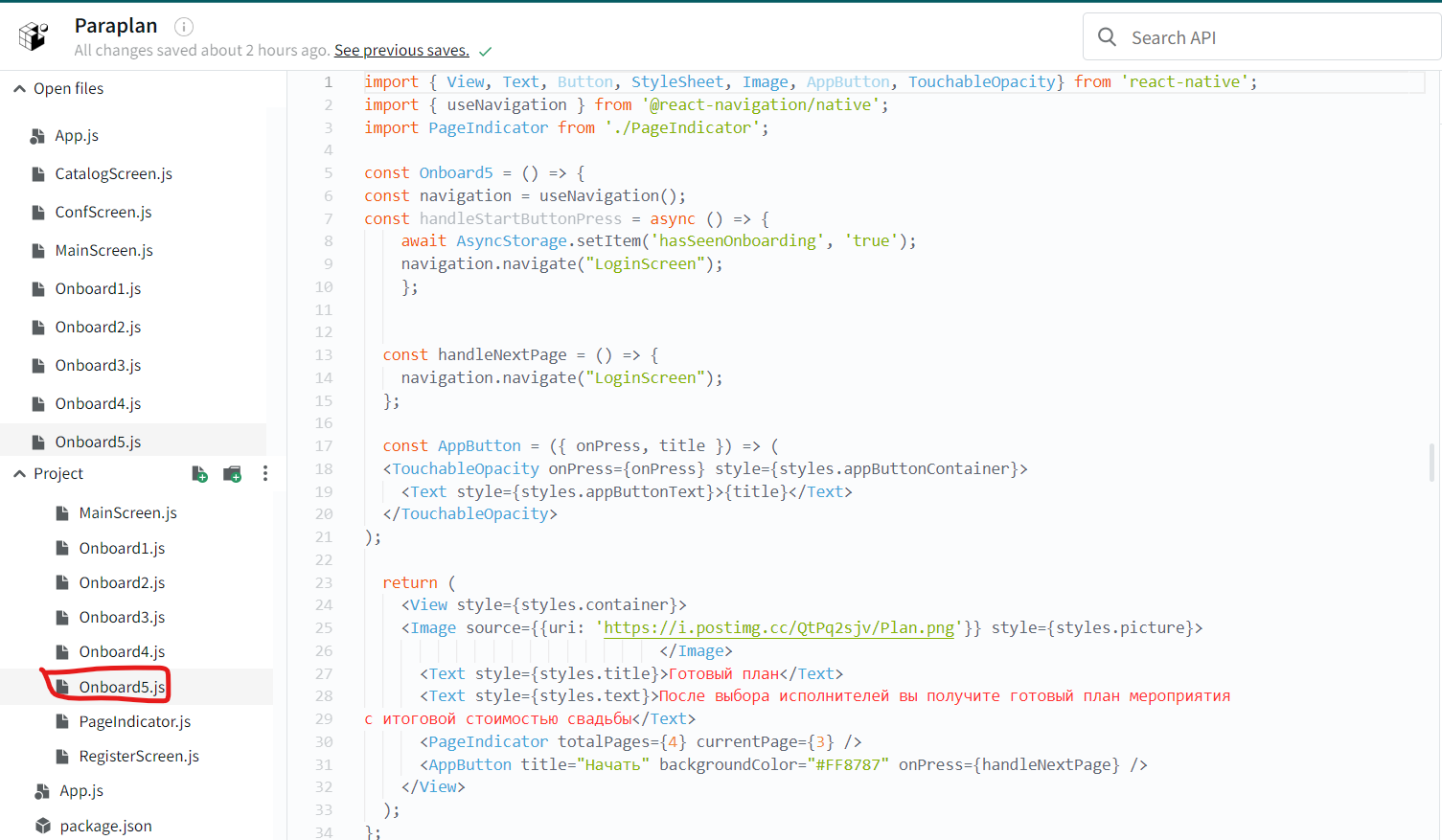
Все пять онбордов добавлены для ознакомления пользователя с информацией. При повторном посещении приложения онборды не будут представлены к просмотру.











# **Итоги проекта**

В заключении проекта, хотелось бы продемонстрировать видео работы мобильного приложения.

https://youtu.be/6XOiW7hurgc