МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет вычислительной техники

Кафедра МОиПЭВМ

Направление подготовки 09.03.04 «Программная инженерия»

БАКАЛАВРСКАЯ РАБОТА

на тему

Мобильное приложение для обучения элементам воднолыжного спорта

Студент _____ Лебедев Алексей Александрович
(подпись, дата) (ФИО полностью)

Руководитель _____ Князев В.Н.
(подпись, дата) (фамилия, инициалы)

Нормоконтролер _____ Такташкин Д.В
(подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от _____№_____)

Заведующий кафедрой _____ Макарычев П.П.
(подпись) (фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от _____№____)

Секретарь ГЭК _____ Попова Н.А.
(подпись) (фамилия, инициалы)

Пенза, 2020

Федеральное государственное бюджетное образовательное учреждение высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет	*Кафедра*
Вычислительной техники	МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
_____П.П. Макарычев
«__»_____ 20___ г

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА

1. Студент Лебедев Алексей Александрович гр. 16ВП1 факультета ВТ
<Фамилия, имя, отчество полностью>

2.Тема работы Мобильное приложение для обучения элементам воднолыжного спорта

Тема утверждена приказом ПГУ №1488/о от "30" декабря 2019

3. Руководитель работы к.т.н., доцент кафедры «МОиПЭВМ» Князев В.Н.

4. Задание на работу (назначение разработки, исходные данные и т.п.)

1) Разработать клиентскую и серверную часть мобильного приложения для обучения элементам воднолыжного спорта, позволяющего пользователю изучать приемы нового вида спорта – вейкбординга.

2) Среда выполнения: операционная система iOS 9.3 и выше для мобильного приложения на IOS, операционная система Android 5.0.3 и выше для мобильного приложения на Android, UNIX-подобная операционная система для серверной части.

Языковые средства разработки: Python, JavaScript, UML.

Инструментальные средства: VS Code, PostgreSQL, Python 2.7, XCode 11, Draw.io, *Microsoft Project, Android Studio, ERwin Data Modeler, эмулятор Iphone 11 Max Pro, эмулятор Pixel PX.*

3) Технологии разработки RUP, ООП, REST API

Основные функции приложения:

а) воспроизведение видео уроков;

б) тестирование пользователя;

в) предоставление полезной информации пользователю;

г) предоставление информации о предстоящих соревнованиях;

д) отображение текущего местоположения вейк станций и информации о них.

5. Перечень подлежащих разработке вопросов:

1) *Анализ предметной области и постановка задачи;*
2) *Проектирование;*
3) *Реализация;*
4) *Контроль качества программного обеспечения;*
5) *Оформление пояснительной записки.*

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	*Анализ предметной области и постановка задачи*	15%	13.04.2020–21.04.2020	
2	*Проектирование*	25%	22.04.2020–05.05.2020	
3	*Реализация*	40%	06.05.2020–30.05.2020	
4	*Контроль качества программного обеспечения*	5%	31.05.2020–02.06.2020	
5	*Оформление пояснительной записки*	15%	03.06.2020–13.06.2020	

Дата выдачи задания " 30" декабря 2019

Руководитель бакалаврской работы _____
(подпись, дата)

Задание к исполнению принял студент _____
(подпись, дата)

Работу к защите допустить
Декан факультета _____
(подпись, дата)

Реферат

Пояснительная записка содержит 161 лист, 36 рисунков, 14 таблиц, 25 использованный источник и 7 приложений.

МОБИЛЬНОЕ ПРИЛОЖЕНИЕ, ОБУЧЕНИЕ, ВЕЙКБОРДИНГ, ANDROID, IOS, RUP, UML, ERWIN DATA MODELER, JAVASCRIPT, XCODE, PYTHON, FALCON, REST API, POSTGRESQL.

Цель выпускной квалификационной работы – разработать серверную и клиентскую часть мобильного приложения для обучения элементам воднолыжного спорта.

В процессе разработки программного продукта применялись технологии RUP, ООП, REST API. Для разработки использовались инструментальные средства Microsoft Project, Draw.io. Проектирование проводилось с использованием языка моделирования UML. Реализация компонентов сервиса проводилась на языках программирования JavaScript и Python в среде разработки Visual Studio Code. Работа с базой данных осуществлялась с помощью СУБД PostgreSQL и технологии Object Relational Mapping.

Мобильное приложение и серверная часть отлажены и протестированы с использованием симулятора смартфонов iPhone 11 Max Pro и Pixel PX.

В результате выполнения выпускной квалификационной работы разработано мобильное приложение для обучения элементам воднолыжного спорта, отвечающее всем требованиям технического задания.

Содержание

Введение 7
1 Анализ предметной области и требований 9
1.1 Анализ предметной области и постановка задачи 9
1.2 Анализ аналогичных программных продуктов 10
1.2.1 Анализ мобильного приложения «Вейкбординг» 10
1.2.2 Анализ мобильного приложения «Learn Wakeboarding Free» 11
1.2.3 Анализ мобильного приложения «WakeTips» 13
1.3 Анализ целевой аудитории 14
1.4 Анализ требований 14
1.4.1 Анализ функциональных требований 15
1.4.2 Анализ нефункциональных требований 23
1.5 Планирование разработки мобильного приложения и оценка бюджета 23
2 Проектирование мобильного приложения 35
2.1 Выбор технологий, инструментальных и языковых средств разработки 35
2.2 Проектирование архитектуры программного обеспечения проекта 36
2.3 Проектирование интерфейса пользователя 37
2.4 Проектирование серверной части приложения 46
2.4.1 Логическое проектирование базы данных 46
2.4.2 Проектирование API 48
3 Реализация мобильного приложения 49
3.1 Реализация серверной части 49
3.1.1 Реализация базы данных 49
3.1.2 Разработка серверной части приложения 50
3.2 Реализация мобильного приложения 51
4 Контроль качества программного продукта 57
4.1 Тестирование серверной части 57
4.2 Тестирование клиентской части приложения 59
4.2.1 Проверка эргономичности пользовательского интерфейса 59
4.2.2 Проверка функциональных возможностей приложения 61
4.2.3 Метрики кода 63
4.2.3.1 Метрика Холстеда 64
4.2.3.2 Метрика уровня комментированности программы 66
Заключение 68
Список использованных источников 69

Приложение А. Глоссарий 72
Приложение Б. Описание атрибутов сущностей базы данных 74
Приложение Г. Исходный код классов серверной части 85
Приложение Д. Исходный код мобильного приложения 123
Приложение Е. Скриншоты запросов. 127
Приложение Ж. Скриншоты юзабилити-тестирования мобильного приложения 142

Введение

В современном мире информационные технологии всё глубже проникают во все отрасли человеческой деятельности. Ярким примером этого может послужить недавний запуск двух систем цифровых пропусков для передвижения по городу в Москве из-за возникшей угрозы распространения новой коронавирусной инфекции. Кроме того, об этом свидетельствует утвержденная в 2017 году правительственная программа «Цифровая экономика Российской Федерации», целью которой является «развитие цифровой экономики Российской Федерации, в которой данные в цифровой форме являются ключевым фактором производства во всех сферах экономической деятельности, что повышает конкурентоспособность страны, качество жизни граждан, обеспечивает экономический рост и экономический суверенитет» [1]. С каждым годом растет спрос на смартфоны, о чем свидетельствует статистика. Так за 2019 год только на территории нашей страны количество проданных смартфонов составляет 30 миллионов единиц. Данные условия свидетельствуют о распространении и доступности мобильных устройств и дают возможность для «стратегии развития информационного общества в Российской Федерации на 2017 – 2030 годы», которая предусматривает использование интернета, веб-сервисов, а также мобильных приложений [2].

В настоящее время занятие спортом становится все популярнее. Кроме того, работа в офисном формате пагубно влияет на здоровье людей. Для поддержания физической формы людям необходимо заниматься активным видом деятельности. Однако, существует категория людей, которым сложно заставить себя делать то, что им не нравится, например, пойти в тренажерный зал или начать бегать. К настоящему моменту существует большое количество активных видов спорта. Одним из таких является вейкбординг. Данный вид спорта не только развивает и укрепляет все группы мышц, но и заставляет почувствовать в себе нотку спортивного азарта, что сделает занятие им еще более интересным и приятным. Вейкбординг является молодым видом спорта,

еще не развитым в нашей стране, поэтому найти достоверную информацию, площадку или вейк станцию, на которой можно заниматься, а также квалифицированного специалиста для изучения элементов воднолыжного спорта, крайне сложно [3].

Наличие мобильного приложения, предоставляющего уроки обучения элементам вейкбординга, полезную информацию о данном виде спорта, месторасположении существующих вейк станций, упростит процесс обучения. Данное приложение поможет изучить новые элементы воднолыжного спорта, предоставит возможность пообщаться с тренером для решения возникших вопросов, сохранить видеоурок на телефон для последующего ознакомления в условиях отсутствия интернета, пройти тестирование после просмотра урока, поможет выбрать оборудование для катания, ознакомит с техникой безопасности, даст советы для начинающих, познакомит с предстоящими соревнованиями, а также подробно покажет месторасположение ближайших вейк станций.

Таким образом, данное приложение может облегчить процесс обучения и поиска вейк станций, а также поможет узнать о данном виде спорта много полезного.

Исходя из вышесказанного, актуальной целью выпускной квалификационной работы является разработка и отладка мобильного приложения, реализуемого для помощи в освоении нового вида спорта – вейкбординга.

1 Анализ предметной области и требований

1.1 Анализ предметной области и постановка задачи

В современном мире все сильнее просматривается мода на занятие спортом. Активный вид деятельности особенно необходим людям, работающим в офисах, потому как сидящий образ жизни пагубно сказывается на здоровье работников. Это чревато развитием сердечно-сосудистых заболеваний, диабета, замедлением метаболизма и даже возможно частичным повреждением головного мозга [4]. На данный момент в мире предоставлено множество возможностей для увеличения активности после офисной работы. Существуют различные тренажерные залы, бассейны, стадионы, спортивные площадки и прочее. Однако имеется категория людей, которым сложно заставить себя заняться тем, что им не нравится, или которым все представленные выше площадки уже наскучили. Возможным решением для них будет занятие вейкбордингом. Данный вид спорта и активного отдыха не только формирует мышечный скелет, но и укрепляет все группы мышц. Кроме того, во время занятия данной дисциплиной появляется спортивный интерес, который будет побуждать человека тренироваться и соревноваться с товарищами в количестве и качестве исполняемых акробатических элементов.

Как и многие спортивные дисциплины, вейкбординг подразделяется на группы. Изначально существовал только вейкбординг с тягой за катером. Однако вскоре стало ясно, что данный вид катания является финансово затратным и не доступен для большой группы людей, так как покупка катера для данной дисциплины в среднем стоит от 9 миллионов рублей, а перед каждым заездом его необходимо заправлять топливом [5]. В связи с этим появился кабельный вейкбординг. Кабельная электротяга симулирует тягу катера, при этом являясь дешевле и удобнее. Катание происходит на одном участке водной поверхности в реверсивном или круговом направлении под присмотром инструктора. На круговой дорожке могут кататься несколько человек, что существенно уменьшит время ожидания в очереди.

Вейкбординг является молодым видом активного спорта, которым занимаются профессиональные спортсмены и любители, взрослые и дети. Несмотря на свою популярность данный вид спорта не только слабо развит в нашей стране, но и мало известен, поэтому человеку, начинающему свою спортивную деятельность, сложно найти квалифицированного специалиста, который смог бы объяснить принципы выполнения элементов, или качественный русскоязычный материал для изучения. Кроме того, площадок для занятия данным видом спорта не так много [6]. Необходимо создать такое приложение, которое поможет человеку с обучением, проведет тестирование изученного материала, изложит основные правила техники безопасности, поможет выбрать оборудование, даст советы начинающему, покажет информацию о предстоящих соревнованиях, а также покажет месторасположение ближайших вейк станций и информацию о них. Приложение должно быть кроссплатформенным и использоваться на мобильных устройствах с такими операционными системами, как Android и IOS.

Вся информация о уроках, соревнованиях и вейкпарках должна быть представлена в виде данных, хранящихся на сервере. Приложение, установленное на мобильном устройстве должно иметь Интернет-соединение для обращения к серверу и включенную геолокацию.

1.2 Анализ аналогичных программных продуктов

В настоящее время существуют аналогичные приложения, созданные для ознакомления с данным видом спорта. Проведем сравнительный анализ наиболее популярных приложения данной тематики.

1.2.1 Анализ мобильного приложения «Вейкбординг»

Вейкбординг – русскоязычное приложение версии 1.0, которое было разработано Decker Publ и выпущено в январе 2015 года. Для работы приложения необходимо мобильное устройство с операционной системой

android версии 4.0.3 или выше [7]. Данное приложение обладает малыми функциональными возможностями. Открыв приложение, пользователь может прочитать статью об основах данного вида спорта: о выборе подходящего снаряжения, о первых шагах начинающего и др.

Главный и единственный экран мобильного приложения представлен на рисунке 1.

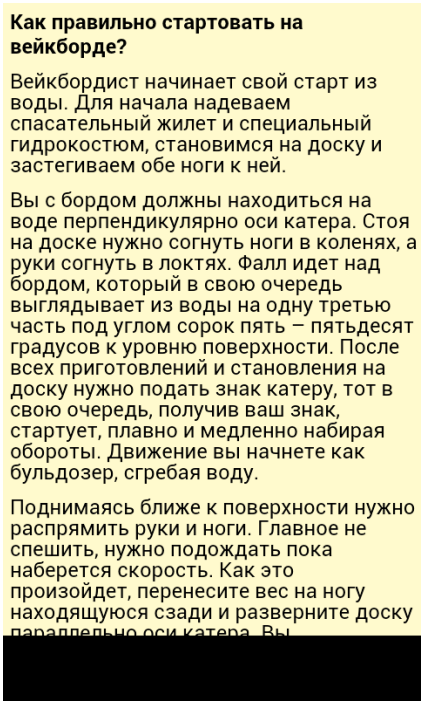


Рисунок 1 – Главный экран мобильного приложения «Вейкбординг»

Среди достоинств данного приложения можно выделить то, что приложение является бесплатным и вся представленная в нем информация на русском языке.

Приложение имеет ряд недостатков: отсутствует навигация, фото и видеоматериалы, что значительно затрудняет усвоение материала. Представленный в приложении текст выглядит не как мобильное приложение, а как статья в интернете. Кроме того, мобильное приложение не является кроссплатформенным.

1.2.2 Анализ мобильного приложения «Learn Wakeboarding Free»

Learn Wakeboarding Free – приложение на английском языке версии 1.0, выпущенное в августе 2017 года. Для работы приложения необходимо

мобильное устройство с мобильной операционной системой Android версии 4.0 или выше [8]. Открыв данное приложение, пользователь имеет возможность выбора интересующего его раздела. Приложение «Learn Wakebording Free» поделено на 10 разделов. В разделах приведено описание экстремального вида спорта Вейкбординга, краткий обзор на виды площадок для катания, инструкция по подбору оборудования, краткое описание акробатических элементов и др.

Главный экран приложения представлен на рисунке 2.

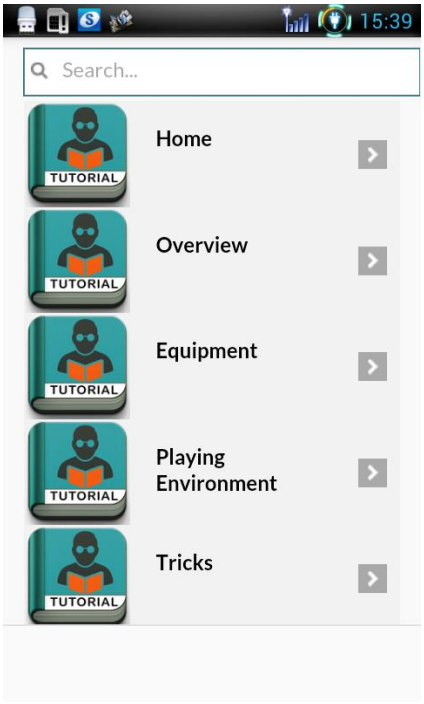


Рисунок 2 – Главный экран мобильного приложения «Learn Wakeboarding Free»

Преимуществом данного приложения является большой объем полезной информации, навигация по разделам и то, что оно является доступным для всех пользователей.

Большим недостатком данного приложения является отсутствие русификации, акробатические элементы описаны сокращено, а также отсутствует видеоматериал, что затрудняет обучение. Также приложение не является кроссплатформенным, что существенно сужает круг пользователей.

1.2.3 Анализ мобильного приложения «WakeTips»

Данное приложение было выпущено в декабре 2013 года David'ом Ngiam. Оно является кроссплатформенным и может быть использовано на мобильных телефонах с операционными системами Android и IOS [9].

Главный экран приложения представлен на рисунке 3.

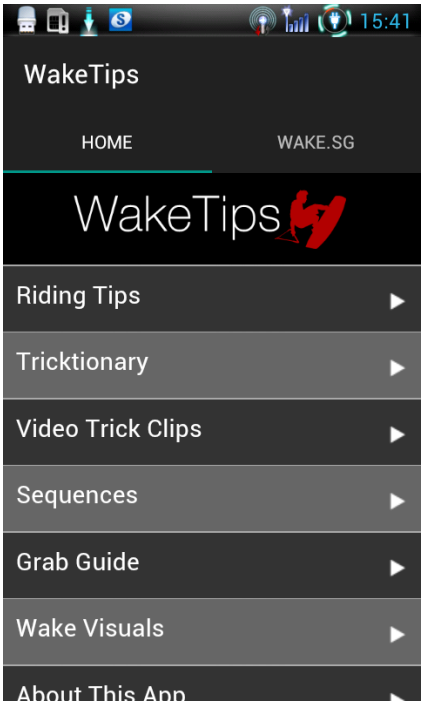


Рисунок 3 – Главный экран мобильного приложения «WakeTips»

Среди преимуществ приложения можно выделить большое количество полезной информации о вейкбординге, присутствует навигация по разделам, а также видеоматериал.

К недостаткам приложения можно отнести отсутствие описания к видеоматериалам, вся приведенная информация на английском языке, не смотря на большое количество видеоматериала, он не подходит для обучения, так как не содержит образовательного подтекста, кроме того приложение платное.

В результате проведенного анализа программ, аналогичных требуемому, было решено создать собственное кроссплатформенное мобильное приложение для обучения элементам воднолыжного спорта, не содержащее выявленных недостатков.

1.3 Анализ целевой аудитории

В настоящее время сложно представить человека без мобильного телефона. О популярности мобильного устройства свидетельствует статистика. Так за 2019 год только на территории Российской Федерации количество проданных смартфонов составило 30 миллионов единиц, а во всем мире количество людей, пользующимися мобильными телефонами, увеличилось на 100 миллионов и составило 5,1 миллиардов человек. Отсюда можно сделать вывод, что мобильные устройства есть у двух третей населения земного шара [10].

Основными пользователями разрабатываемого приложения будут люди, желающие заняться активным экстремальным видом спорта. Одним из таких является вид воднолыжного спорта – вейкбординг. Рост популярности вейкбординга можно отследить по статистике открывающихся вейкпарков. На данный момент только в Европе количество таких парков превысило отметку равную 100 [11]. Кроме того, известность вейкбординга увеличивает Международный Олимпийский Комитет (МОК), который включил данную дисциплину в шорт-лист видов спорта, претендующих на включение в программу Летних Олимпийских игр 2020 года [12]. Исходя из сказанного выше, можно сделать вывод, что популярность вейкбординга растет, что в свою очередь приводит к увеличению людей, стремящихся освоить данный вид спорта.

Основными мобильные операционными система на данный момент остаются Android и IOS. Следовательно, для обучения необходимо кроссплатформенное мобильное приложение, позволяющее пользователям Android и IOS устройств использовать весь обучающий и информационный функционал программы.

1.4 Анализ требований

На этапе анализа требований происходит сбор, структуризация требований и документирование. Цель данного этапа заключается в

предоставлении четкого списка не дублируемых требований к системе. Правильно сгруппированные требования помогут обойтись минимальным количеством функционала для удовлетворения большего количества целей, что поможет сэкономить бюджет.

Требования принято разделять на функциональные и нефункциональные. Функциональные требования описывают поведение программной системы, нефункциональные требования объясняют свойства приложения.

1.4.1 Анализ функциональных требований

В результате анализа предметной области и аналогичных программных продуктов были выделены следующие функциональные требования к разрабатываемому приложению:

- посмотреть информацию о тренере;
- задать тренеру вопрос;
- посмотреть youtube канал тренера;
- просмотреть список видеоуроков;
- посмотреть видеоурок;
- скачать видеоурок на мобильное устройство;
- пройти тестирование;
- посмотреть полезную информацию;
- прочитать историю;
- прочитать советы для начинающих;
- ознакомиться с техникой безопасности;
- узнать о выборе оборудования;
- просмотреть список соревнований;
- посмотреть информацию о предстоящих соревнованиях;
- зарегистрироваться в соревнованиях;
- посмотреть месторасположение ближайших вейк станций;
- посмотреть информацию о выбранной вейк станции;

– позвонить по контактному номеру.

Диаграмма вариантов использования для мобильного приложения представлена на рисунке 4.

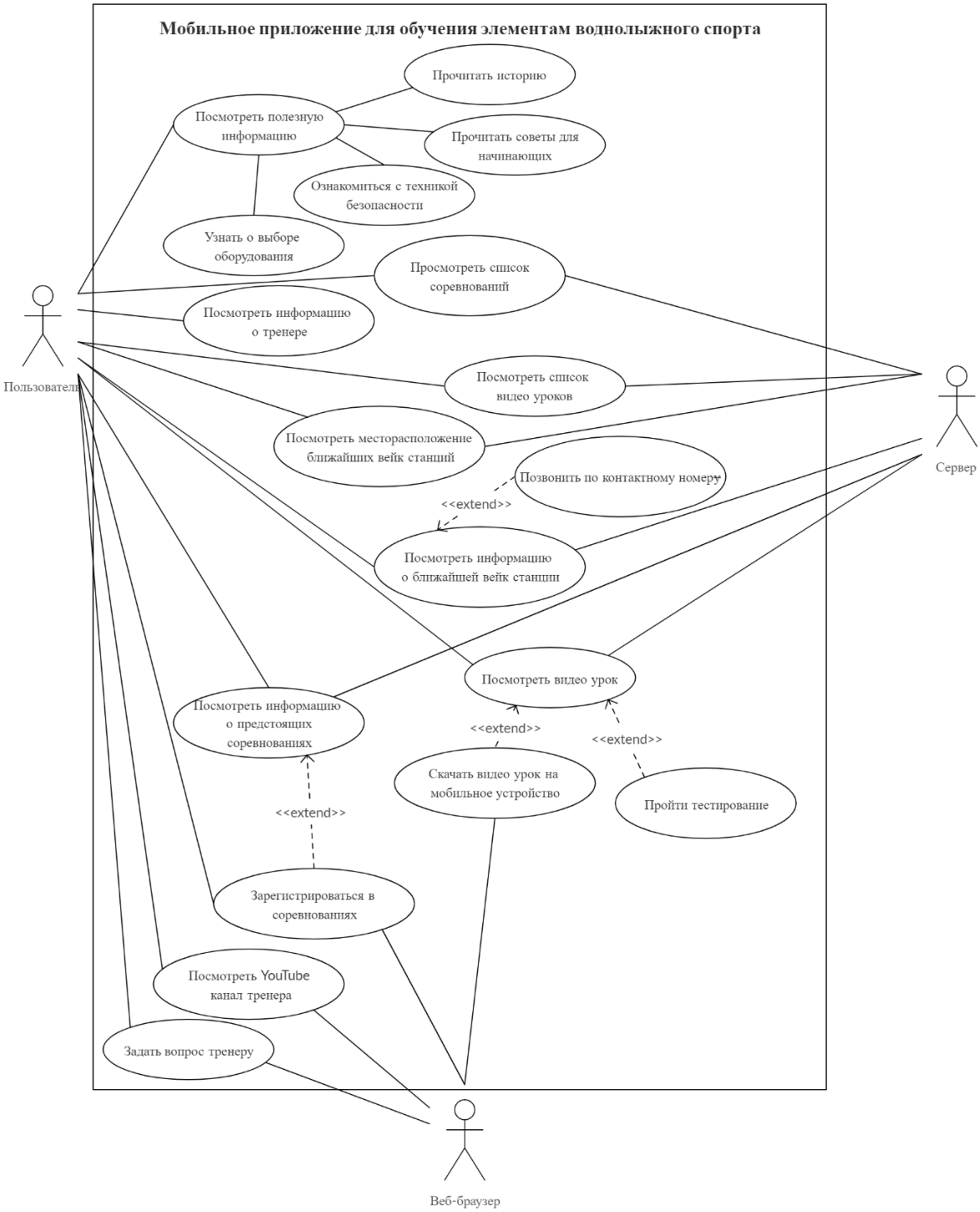


Рисунок 4 – Диаграмма вариантов использования мобильного приложения «WakeboardTeacher»

Приведем подробную спецификацию некоторых прецедентов. На рисунке 5 изображена спецификация варианта использования «Посмотреть видео урок».

Прецедент: Просмотреть видео урок
ID: 1
Краткое описание: Пользователь выбирает необходимый ему видео урок в списке и просматривает его.
Главные актёры: Пользователь
Второстепенные актёры: Сервер
Предусловия: Пользователь должен находиться на вкладке «Уроки» в панели навигации в нижней части экрана
Основной поток: 1. Пользователь выбирает нужный ему урок в списке. 2. Сервер отправляет данные выбранного видео урока. 3. Пользователь переходит на страницу данного урока.
Постусловия: нет.
Альтернативные потоки: нет.

Рисунок 5 – Спецификация варианта использования «Просмотреть видео урок»

Проводя сравнение с аналогичными мобильными приложениями, можно сделать вывод, что данный вариант использования в них развит слабо, либо вовсе отсутствует. Для хорошего усваивания материала пользователю необходимо предоставить видео уроки на русском языке, а также добавить список подводящих упражнений и описание самых важных моментов, которые помогут обучающемуся изучить элемент с минимальным количеством ошибок, что существенно сократит число возможных травм. Дальнейшая разработка данного прецедента будет опираться на указанные требования.

На рисунке 6 представлена спецификация варианта использования «Посмотреть информацию о вейк станции».

Прецедент: Посмотреть информацию о вейк станции
ID: 2
Краткое описание: Пользователь нажимает на маркер, отображающий интересующую вейк станцию, и смотрит информацию о вейк станции
Главные актёры: Пользователь
Второстепенные актёры: Сервер
Предусловия: Пользователь должен находиться на вкладке «Вейкпарки» в панели навигации в нижней части экрана
Основной поток: 1. Пользователь нажимает на маркер, отображающий интересующий его вейк станцию. 2. Сервер отправляет данные выбранной вейк станции. 3. Мобильное приложение отображает полученные данные. 4. Пользователь просматривает полученную информацию.
Постусловия: нет.
Альтернативные потоки: нет.

Рисунок 6 – Спецификация варианта использования «Посмотреть информацию о вейк станции»

Данный вариант использования отсутствует во все приведенных приложениях-аналогах. Информация о вейк станции поможет пользователю выбрать ближайшую площадку для занятий, огласив стоимость катания за один раз (сет) или за весь день катания, что позволит пользователю выбрать станцию с самыми низкими ценами, адрес вейк парка, а также контактный телефон, по которому пользователь сможет связаться с представителем вейк станции для уточнения деталей.

На рисунке 7 изображена спецификация для варианта использования «Прочитать советы для начинающих».

Прецедент: Прочитать советы для начинающих
ID: 3
Краткое описание: Пользователь знакомится с советами для начинающих.
Главные актёры: Пользователь
Второстепенные актёры: нет.
Предусловия: Пользователь должен находиться на вкладке «Вейкбординг» в панели навигации в нижней части экрана.
Основной поток: 1. Пользователь выбирает раздел «Советы для начинающих». 2. Пользователь знакомиться с советами для начинающих.
Постусловия: нет.
Альтернативные потоки: нет.

Рисунок 7 – Спецификация варианта использования «Прочитать советы для начинающих»

Данный вариант использования приведен в приложении-аналоге «Вейкбординг». Однако из-за избыточного количества неструктурированного материала у пользователя могут возникнуть проблемы с усвоением при изучении. Необходимо разбить материал на разделы и кратко, но максимально развернуто описать основные моменты, которые помогут начинающим освоить данный вид спорта, поведают историю создания вейкбординга, ознакомят с техникой безопасности и помогут новичку с выбором его первого персонального оборудования.

Для спецификации исполняемого поведения в виде координированного последовательного и параллельного выполнения подчинённых элементов, соединенных между собой потоками составим диаграммы деятельности.

На рисунке 8 приведена диаграмма деятельности для прецедента «Просмотреть список соревнований».

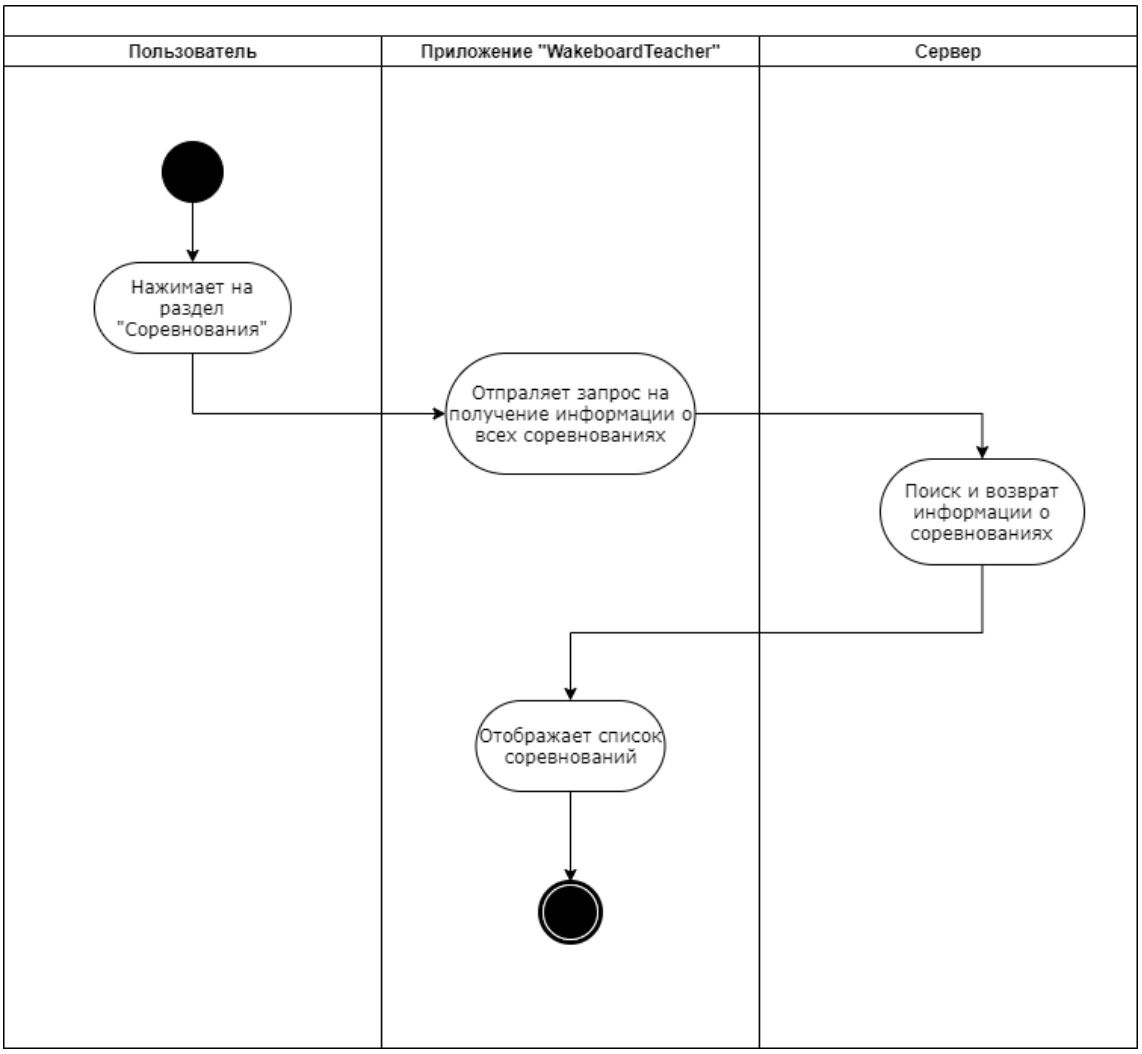


Рисунок 8 – Диаграмма деятельности для варианта использования «Просмотреть список соревнований»

Представленная на рисунке 8 диаграмма деятельности изображается в форме графа, вершинами которого являются состояния действия, а дугами – переходы от одного действия к другому. Данная диаграмма содержит 3 раздела, отображающие актеров системы: пользователь, приложение «WakeboardTeacher» и сервер. Для получения списка соревнований пользователь должен перейти в раздел «Соревнования». Мобильное приложение делает запрос на сервер к базе данных. Сервер возвращает информацию о предстоящих состязаниях и приложение динамически формирует список из полученных с сервера данных.

На рисунке 9 приведена диаграмма деятельности для прецедента «Скачать видео урок на мобильное устройство».

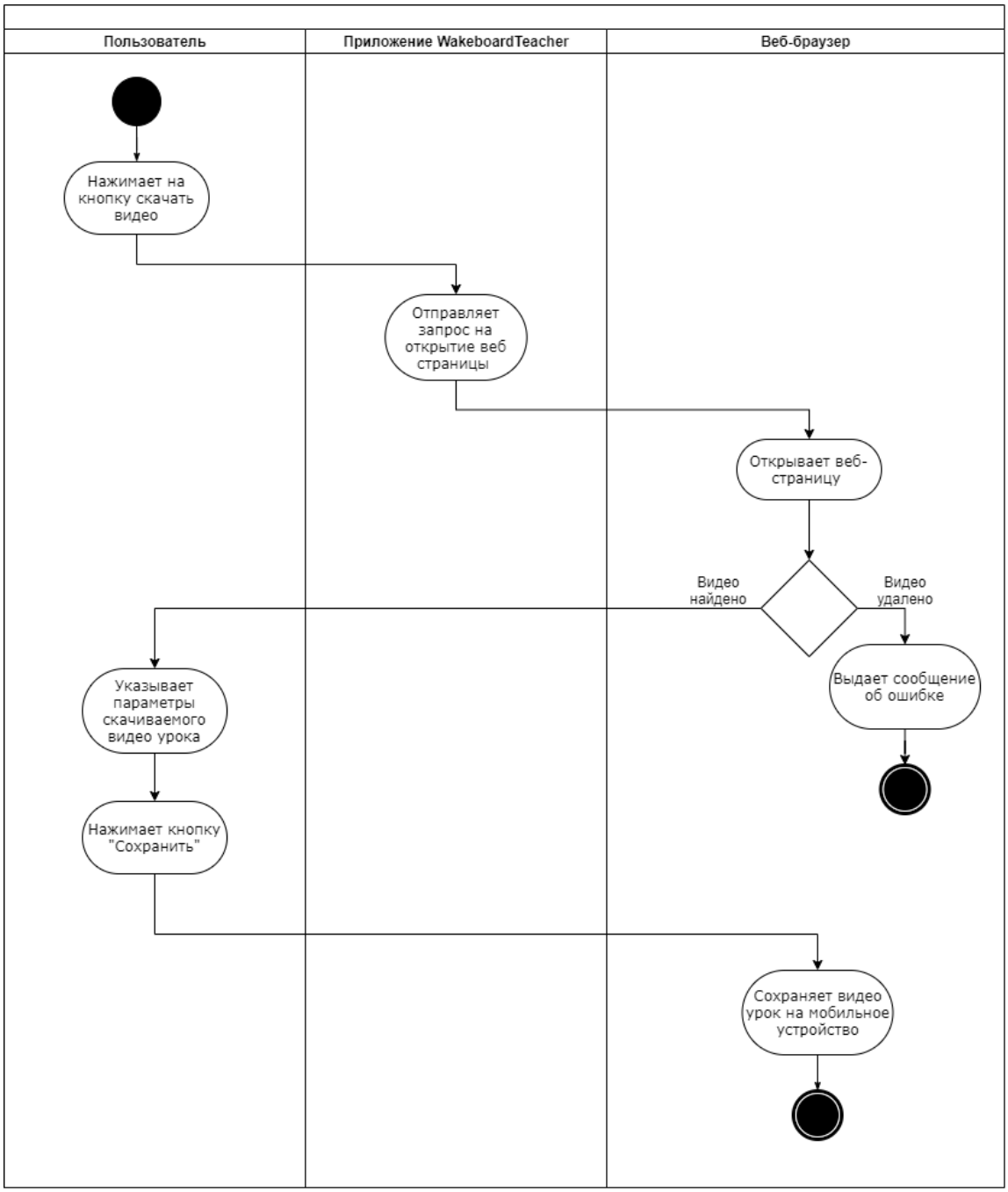


Рисунок 9 – Диаграмма деятельности для варианта использования «Скачать видео урок на мобильное устройство»

Для варианта использования «Посмотреть месторасположение ближайших вейк станций» создадим диаграмму последовательности, которая будет отображать участвующих во взаимодействии актеров и последовательность их общения. Она представлена на рисунке 10.

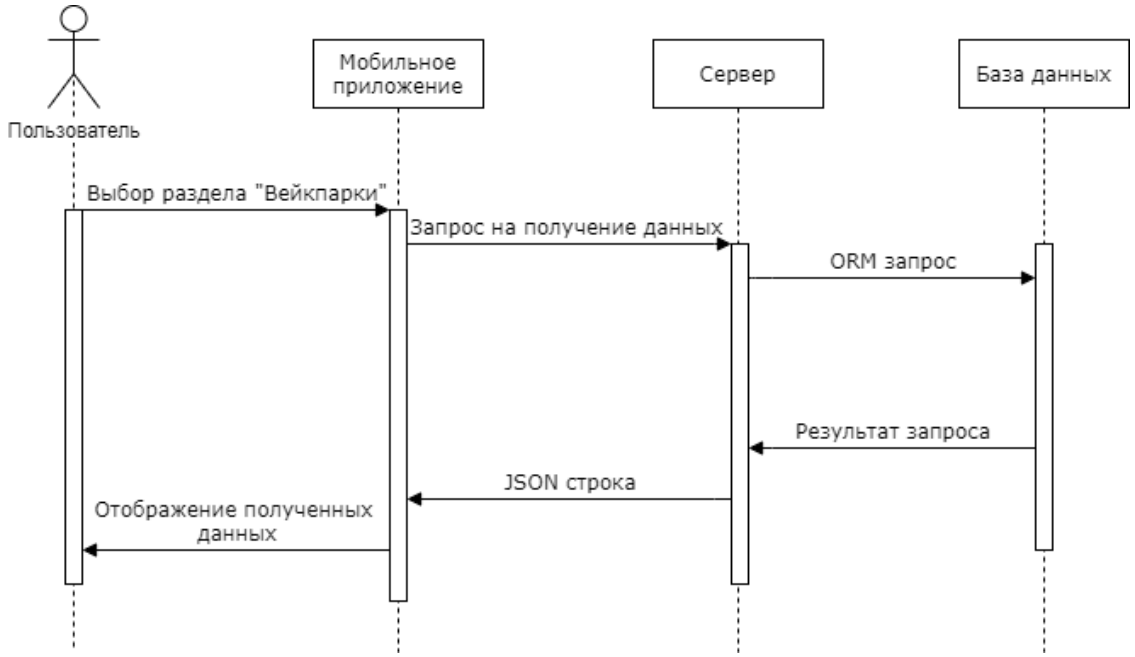


Рисунок 10 – Диаграмма последовательности для варианта использования «Посмотреть месторасположение ближайших вейк станций»

На рисунке 11 представлена диаграмма последовательности для варианта использования «Пройти тестирование».

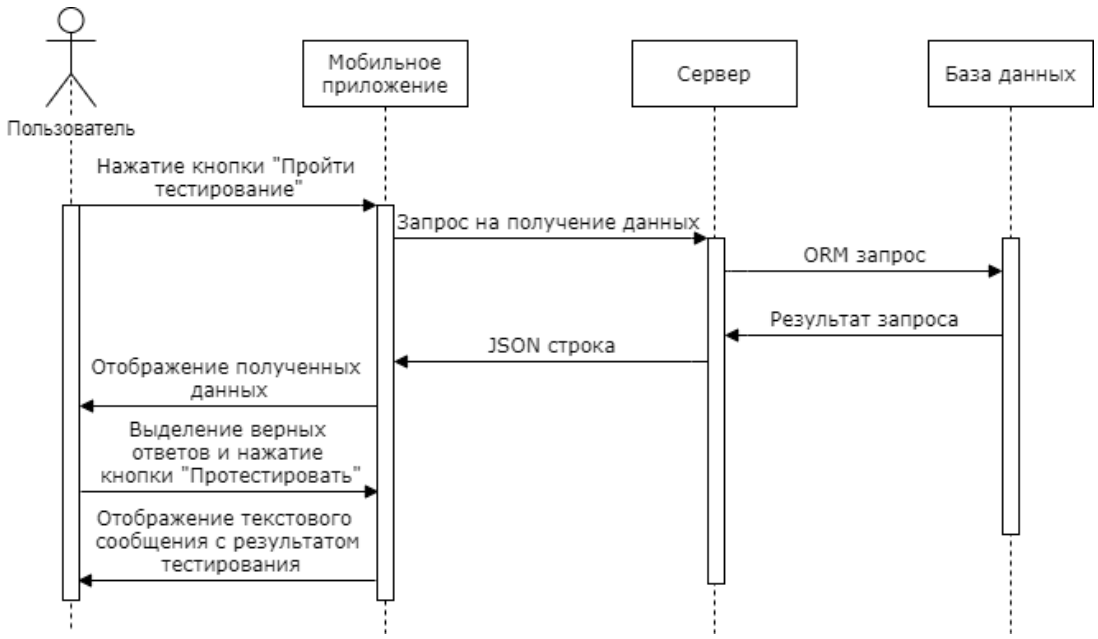


Рисунок 11 – Диаграмма последовательности для варианта использования «Пройти тестирование»

В результате проведенного анализа функциональных требований были сформированы основные функции, которые необходимо реализовать в

разрабатываемом программном продукте, а также была составлена диаграмма вариантов использования, разработаны спецификации и построены диаграммы деятельности для некоторых прецедентов разрабатываемой программы.

1.4.2 Анализ нефункциональных требований

Для разработки кроссплатформенного приложения выбран молодой, но многообещающий фреймворк React Native. Реализация клиентской части приложения должна быть выполнена на языке JavaScript и при помощи компонентов фреймворка. Разработка серверной части приложения должна быть произведена на языке Python и веб-среды Falcon. В качестве СУБД была выбрана PostgreSQL.

Клиентская часть приложения должна быть отлажена и работать одинаково качественно на мобильных операционных системах Android и IOS.

1.5 Планирование разработки мобильного приложения и оценка бюджета

Для определения трудоемкости и стоимости проекта необходимо провести его грамотное планирование. Для этого были определены список задач, решение которых необходимо для реализации приложения, и перечень ресурсов, с помощью которых эти задачи будут решаться.

В таблице 1 представлены задачи, представляющие из себя этапы создания проекта.

Таблица 1 – Перечень задач проекта

№	Название задачи	Длительность	Предшественники
1	Начало проекта	-	-
2	Анализ требований	-	-
3	Поиск приложений-аналогов	2 дня	1
4	Поиск литературы (методических пособий по воднолыжному спорту)	1 день	3

Продолжение таблицы 1

5	Создание списка функциональных возможностей приложения	3 дня	3
6	Анализ списка функциональных возможностей приложения	2 дня	4;5
7	Консультация с инструкторами воднолыжного спорта	1 день	6
8	Анализ требований завершен	-	7
9	Выбор средств реализации проекта	-	-
10	Анализ возможных программных средств	4 дня	1
11	Выбор программного средства для реализации	2 дня	10
12	Выбор средств реализации завершен	-	8;11
13	Проектирование		
14	Проектирование интерфейса клиентского приложения	5 дней	12
15	Проектирование базы данных	3 дня	12
16	Выбор архитектуры программного обеспечения	3 дня	15
17	Проектирование завершено		14;16
18	Реализация проекта	-	-
19	Создание базы данных	5 дней	17
20	Программирование классов приложения	9 дней	19
21	Интеграция клиентской и серверной части приложения	4 дня	20
22	Реализация проекта завершена	-	21
23	Отладка	-	-
24	Тестирование работы приложения	5 дней	22

Окончание таблицы 1

25	Исправление ошибок	6 дней	22
26	Отладка завершена	-	24;25
27	Конец проекта	-	26

Из этой таблицы можно сделать вывод, что для реализации программного проекта нужно пройти через решение 27 задач.

В таблице 2 представлен список затрачиваемых ресурсов.

Таблица 2 – Список ресурсов проекта

Название ресурса	Тип	Стандартная ставка	Ставка сверхурочных
Программист1	Трудовой	50 Р\ч	50 Р\час
Программист2	Трудовой	50 Р\ч	50 Р\час
Интернет	Затраты		
Компьютер	Материальны	200 Р	
Мобильное устройство	Материальный	200 Р	

Распределение ресурсов, представленных в таблице 2, приведено в таблице 3.

Таблица 3 – Назначения ресурсов

№	Название задачи	Ресурс	Единицы
1	Начало проекта		
2	Анализ требований		
3	Поиск приложение-аналогов	Программист 2 Интернет Компьютер	100% 1 шт.
4	Поиск литературы (методических пособий по воднолыжному спорту)	Программист 2 Интернет Компьютер	100% 1 шт.
5	Создание списка функциональных возможностей приложения	Программист 1 Компьютер	100% 1 шт.

Продолжение таблицы 3

6	Анализ списка функциональных возможностей приложения	Программист 1 Программист 2 Интернет Компьютер	100% 100% 2 шт.
7	Консультация с инструкторами воднолыжного спорта	Программист 1 Интернет Компьютер	100% 1 шт.
8	Анализ требований завершен		
9	Выбор средств реализации проекта		
10	Анализ возможных программных средств	Программист 2 Интернет Компьютер	100% 1 шт.
11	Выбор программного средства для реализации	Программист 1 Программист 2 Интернет Компьютер	100% 100% 2 шт.
12	Выбор средств реализации завершен		
13	Проектирование		
14	Проектирование интерфейса клиентского приложения	Программист 2 Интернет Компьютер	100% 1 шт.
15	Проектирование базы данных	Программист 1 Интернет Компьютер	100% 1 шт.
16	Выбор архитектуры программного обеспечения	Программист 1 Интернет Компьютер	100% 1 шт.
17	Проектирование завершено		
18	Реализация проекта		

Окончание таблицы 3

19	Создание базы данных	Программист 1 Интернет Компьютер	100% 1 шт.
20	Программирование классов приложения	Программист 1 Программист 2 Интернет Компьютер	100% 100% 2 шт.
21	Интеграция клиентской и серверной части приложения	Программист 1 Программист 2 Интернет Компьютер Мобильное устройство	100% 100% 2 шт. 1 шт.
22	Реализация проекта завершена		
23	Отладка		
24	Тестирование работы приложения	Программист 2 Интернет Мобильное устройство	100% 1 шт.
25	Исправление ошибок	Программист 1 Интернет Компьютер Мобильное устройство	100% 1 шт. 1 шт.
26	Отладка завершена		
27	Конец проекта		

Для решения поставленных задач необходимо иметь штат из 2 программистов, 2 компьютера, доступ к интернету и мобильному устройству.

Для начала необходимо посчитать время, необходимое для реализации проекта и определить критический путь. Для этого необходимо построить сетевой график проекта. Он представлен на рисунке 12.

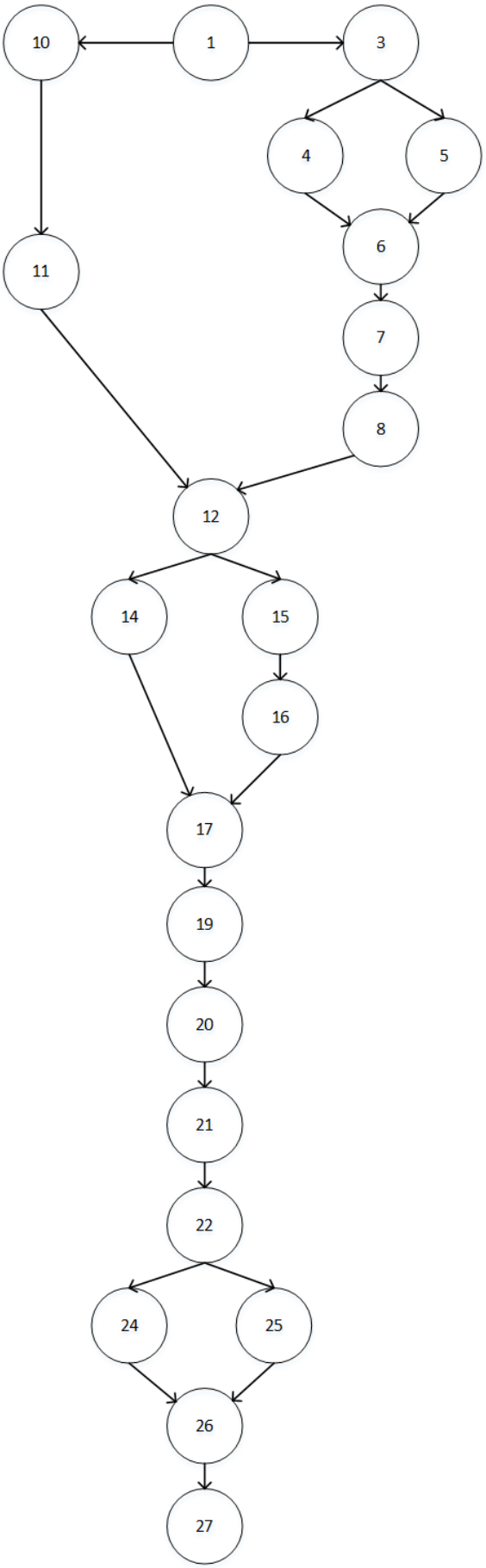


Рисунок 12 – Сетевой график проекта

Обозначим через t_i время выполнения работы i, G – множество работ, непосредственно предшествующих работе i, H – множество работ, непосредственно следующих за работой i.

Для вычисления критического пути необходимо для каждой задачи определить раннее и позднее время начала работ. Ранее время определяется по формуле:

$$T_{\text{р}(i)} = \max_{j \in G}\left(T_{\text{р}(j)} + t_j\right).$$

Позднее время начала следует вычислять по формуле:

$$T_{\text{п}(i)} = \min_{j \in H}\left(T_{\text{п}(j)} - t_j\right).$$

Вычисление раннего времени начала работ расписано в таблице 4.

Таблица 4 – Вычисление раннего времени начала работ

Работа	Предшественники			Вычисление	Раннее время начала
	№	Раннее время начала	Длительность		
1	-	-	-	-	0
3	1	0	0	0+0	0
4	3	0	2	0+2	2
5	3	0	2	0+2	2
6	4 5	2 2	1 3	Max(2+1;2+3)	5
7	6	5	2	5+2	7
8	7	7	1	7+1	8
10	1	0	0	0+0	0
11	10	0	4	0+4	4
12	8 11	8 4	0 2	Max(8+0;4+2)	8
14	12	8	0	8+0	8
15	12	8	0	8+0	8
16	15	8	3	8+3	11
17	14 16	8 11	5 3	Max(8+5;11+3)	14

Продолжение таблицы 4

19	17	14	0	14+0	14
20	19	14	5	14+5	19
21	20	19	9	19+9	28
22	21	28	4	28+4	32
24	22	32	0	32+0	32
25	22	32	0	32+0	32
26	24 25	32 32	5 6	Max(32+5;32+6)	38
27	26	38	0	38+0	38

Таким образом, можно увидеть, что для выполнения проекта понадобиться 38 рабочих дней. Теперь необходимо вычислить время позднего начала работ. Оно представлено в таблице 5.

Таблица 5 – Вычисление позднего времени начала работ

Работа	Последователи		Длительность	Вычисление	Позднее время начала
	№	Позднее время начала			
27	-	-	-	-	38
26	27	38	0	38-0	38
25	26	38	6	38-6	32
24	26	38	5	38-5	33
22	24 25	33 32	0 0	Min(33-0;32-0)	32
21	22	32	4	32-4	28
20	21	28	9	28-9	19
19	20	19	5	19-5	14
17	19	14	0	14-0	14
16	17	14	3	14-3	11
15	16	11	3	11-3	8
14	17	14	5	14-5	9
12	15 14	8 9	0 0	Min(8-0;9-0)	8
11	12	8	2	8-2	6

Продолжение таблицы 5

10	11	6	4	6-4	2
8	12	8	0	8-0	8
7	8	8	1	8-1	7
6	7	7	2	7-2	5
5	6	5	3	5-3	2
4	6	5	1	5-1	4
3	5 4	2 4	2 2	Min(2-2;4-2)	0
1	10 3	2 0	0 0	Min(2-0;0-0)	0

Теперь можно вычислить критический путь для нашего проекта. Вычисления представлены в таблице 6.

Таблица 6 – Вычисление критического пути

Работа №	Раннее время начала	Позднее время начала	Резерв времени
1	0	0	0
3	0	0	0
4	2	4	2
5	2	2	0
6	5	5	0
7	7	7	0
8	8	8	0
10	0	2	2
11	4	6	2
12	8	8	0
14	8	9	1
15	8	8	0
16	11	11	0
17	14	14	0
19	14	14	0
20	19	19	0
21	28	28	0

Продолжение таблицы 6

22	32	32	0
24	32	33	1
25	32	32	0
26	38	38	0
27	38	38	0

Из таблицы видно, что критический путь проходит через работы, с номерами 1, 3, 5, 6, 7, 8, 12, 15, 16, 17, 19, 20, 21, 22, 25, 26, 27. Его можно наглядно проследить на рисунке 13.

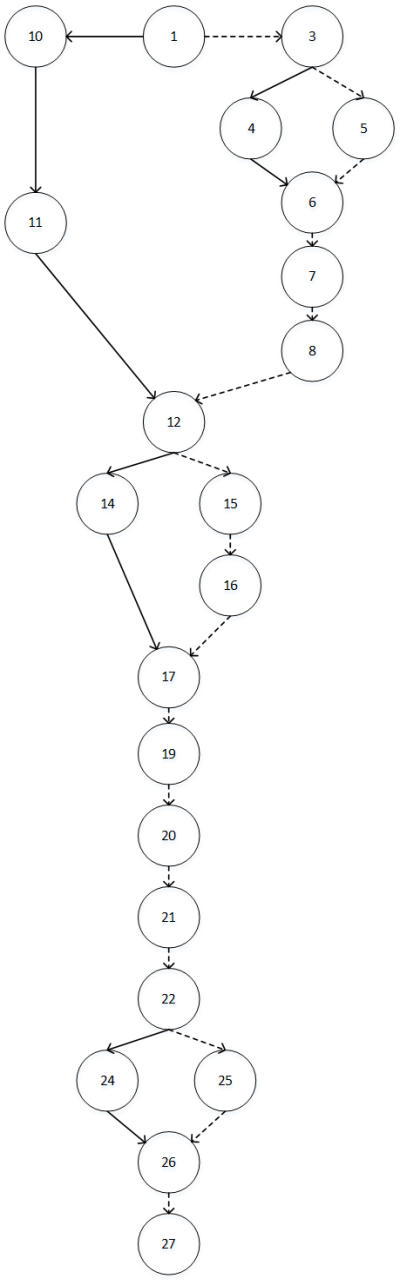


Рисунок 13 – Критический путь

Теперь необходимо внести перечисленные выше данные в систему управления проектами Microsoft Project. На рисунке 14 изображено введенное в Project содержание таблицы 1 с вычисленными датами начала и завершения каждой задачи. Дата начала проекта была назначена на 16.03.2020.

	Название задачи	Длительность	Начало	Окончание	Предшес	Названия ресурсов
1	Начало проекта	0 дней	Пн 16.03.20	Пн 16.03.20		
2	**Анализ требований**	**7 дней**	**Пн 16.03.20**	**Вт 24.03.20**		
3	Поиск приложений-аналогов	2 дней	Пн 16.03.20	Вт 17.03.20	1	Программист 2;Интернет;Компь
4	Поиск литературы (методиче	1 день	Ср 18.03.20	Ср 18.03.20	3	Программист 2;Интернет;Компь
5	Создание списка функционал	3 дней	Ср 18.03.20	Пт 20.03.20	3	Программист 1;Компьютер [1]
6	Анализ списка функциональ	1 день	Пн 23.03.20	Пн 23.03.20	5;4	Программист 1;Программист 2;
7	Консультация с инструктора	1 день	Вт 24.03.20	Вт 24.03.20	6	Программист 1;Интернет;Компь
8	Анализ требований завершен	0 дней	Вт 24.03.20	Вт 24.03.20	7	
9	**Выбор средств реализации пр**	**7 дней**	**Пн 16.03.20**	**Вт 24.03.20**		
10	Анализ возможных программ	4 дней	Пн 16.03.20	Чт 19.03.20	1	Программист 2;Интернет;Компь
11	Выбор программного средст	2 дней	Пт 20.03.20	Пн 23.03.20	10	Программист 1;Программист 2;
12	Выбор средств реализации з	0 дней	Вт 24.03.20	Вт 24.03.20	11;8	
13	**Проектирование**	**6 дней**	**Ср 25.03.20**	**Ср 01.04.20**		
14	Проектирование интерфейса	5 дней	Ср 25.03.20	Вт 31.03.20	12	Программист 2;Интернет;Компь
15	Проектирование базы данны	3 дней	Ср 25.03.20	Пт 27.03.20	12	Программист 1;Интернет;Компь
16	Выбор архитектуры програм	3 дней	Пн 30.03.20	Ср 01.04.20	15	Программист 1;Интернет;Компь
17	Проектирование завершено	0 дней	Ср 01.04.20	Ср 01.04.20	16;14	
18	**Реализация проекта**	**18 дней**	**Чт 02.04.20**	**Пн 27.04.20**		
19	Создание базы данных	5 дней	Чт 02.04.20	Ср 08.04.20	17	Программист 1;Интернет;Компь
20	Программирование классов	9 дней	Чт 09.04.20	Вт 21.04.20	19	Программист 1;Программист 2;
21	Интеграция клиентской и сер	4 дней	Ср 22.04.20	Пн 27.04.20	20	Программист 1;Программист 2;
22	Реализация проекта заверше	0 дней	Пн 27.04.20	Пн 27.04.20	21	
23	**Отладка**	**6 дней**	**Вт 28.04.20**	**Вт 05.05.20**		
24	Тестирование работы прило	5 дней	Вт 28.04.20	Пн 04.05.20	22	Программист 2;Интернет;Andro
25	Исправление ошибок	6 дней	Вт 28.04.20	Вт 05.05.20	22	Программист 1;Интернет;Компь
26	Отладка завершена	0 дней	Вт 05.05.20	Вт 05.05.20	25;24	
27	Конец проекта	0 дней	Вт 05.05.20	Вт 05.05.20	26	

Рисунок 14 – Задачи проекта, введенные в среду Microsoft Project

На рисунке 15 представлена построенная программой диаграмма Ганта.

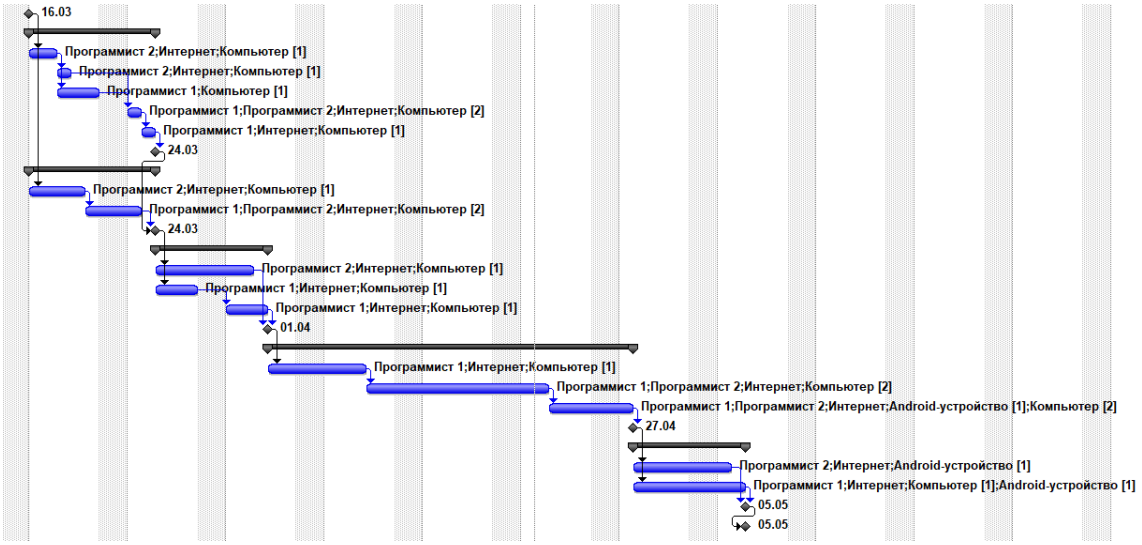
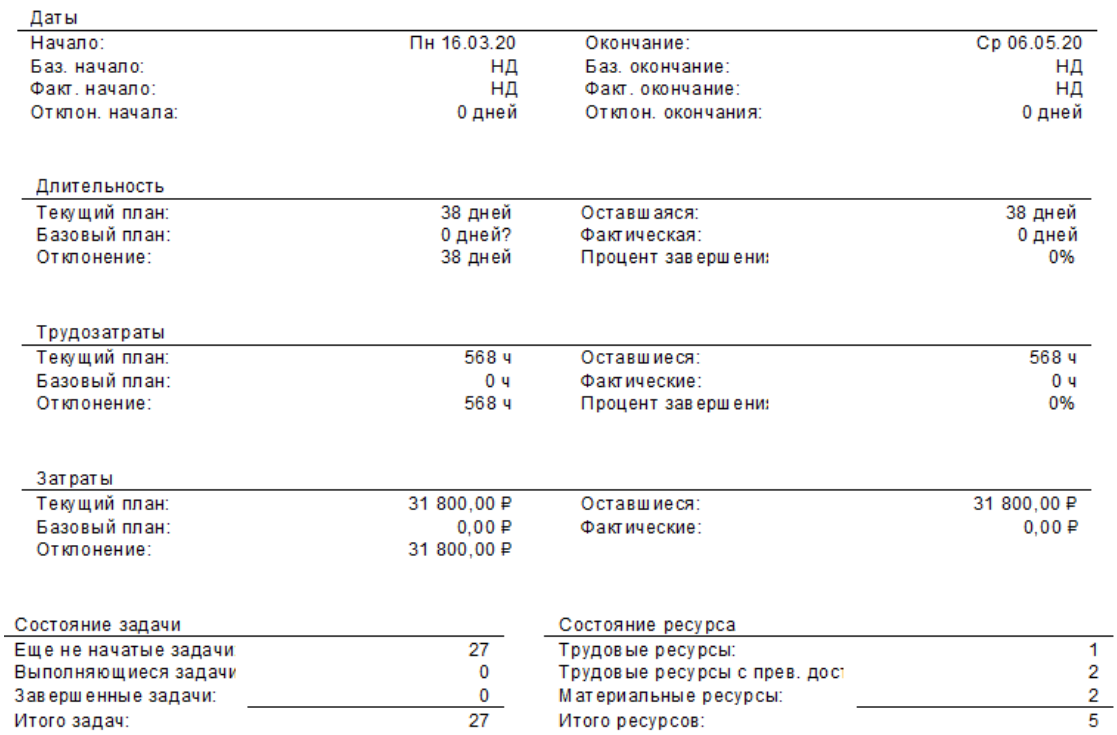


Рисунок 15 – Диаграмма Ганта

Из этих рисунков можно понять, что вычисленный программой критический путь проходит по задачам 1, 3, 5, 6, 7, 8, 12, 15, 16, 17, 19, 20, 21, 22, 25, 26, 27. Из этого можно сделать вывод, что построенный программой график и определенный ею критический путь совпадают с теми, что были вычислены вручную.

На рисунке 16 изображен обзор проекта, отображающий общую стоимость и продолжительность проекта.



Даты			
Начало:	Пн 16.03.20	Окончание:	Ср 06.05.20
Баз. начало:	НД	Баз. окончание:	НД
Факт. начало:	НД	Факт. окончание:	НД
Отклон. начала:	0 дней	Отклон. окончания:	0 дней

Длительность			
Текущий план:	38 дней	Оставшаяся:	38 дней
Базовый план:	0 дней?	Фактическая:	0 дней
Отклонение:	38 дней	Процент завершени:	0%

Трудозатраты			
Текущий план:	568 ч	Оставшиеся:	568 ч
Базовый план:	0 ч	Фактические:	0 ч
Отклонение:	568 ч	Процент завершени:	0%

Затраты			
Текущий план:	31 800,00 ₽	Оставшиеся:	31 800,00 ₽
Базовый план:	0,00 ₽	Фактические:	0,00 ₽
Отклонение:	31 800,00 ₽		

Состояние задачи	
Еще не начатые задачи	27
Выполняющиеся задачи	0
Завершенные задачи:	0
Итого задач:	27

Состояние ресурса	
Трудовые ресурсы:	1
Трудовые ресурсы с прев. дос	2
Материальные ресурсы:	2
Итого ресурсов:	5

Рисунок 16 – Обзор проекта

Обзор проекта показал подсчитанную Microsoft Project длительность реализации проекта – 38 дней (что совпадает с вычисленной ранее вручную), и предполагаемые затраты – 31 800 руб.

2 Проектирование мобильного приложения

2.1 Выбор технологий, инструментальных и языковых средств разработки

Выбранная методология разработки Rational Unified Process (RUP) использует итеративный подход к реализации программного продукта. Весь проект разбивается на совокупность небольших проектов, выполняющихся последовательно. В конце каждой итерации должны быть решены задачи и цели, выделенные для данной итерации, и получена промежуточная функциональная версия программного изделия. Данный тип разработки позволяет обнаружить ошибки на ранних этапах разработки и быстро отреагировать на меняющиеся требования [13].

Языком моделирования методологии RUP является Unified Modelling Language (UML). Данный язык используется для объектно-ориентированного анализа и построения диаграмм, моделирующих бизнес-процессы и структуры программного продукта [14].

Рассмотренные достоинства Rational Unified Process позволяют выбрать данную методологию для разработки мобильного приложения, обучающего элементам воднолыжного спорта.

На этапе планирования разработки мобильного приложения была проведена оценка бюджета с помощью программного средства Microsoft Project семейства Microsoft Office, что позволило создать план, проанализировать объем работ, распределить время и ресурсы для реализации поставленных задач.

Создание мобильного приложения должна производиться с помощью фреймворка React Native, предназначенного для кроссплатформенной разработки под Android и IOS [15]. Для реализации и отладки приложения под мобильную операционную систему IOS должна использоваться среда разработки XCode 11 со встроенным эмулятором Iphone 11 Pro, установленная на MacBook, и язык программирования JavaScript. Для отладки программного

продукта под Android следует использовать Visual Studio Code, язык программирования JavaScript, эмулятор Pixel PX встроенный в Android Studio, установленные на операционную систему Ubuntu 18.04.

Серверная часть программного продукта должна представлять собой REST API для клиентских приложений и выполняет взаимодействие с базой данных. Создание REST API должно производиться на фреймворке Falcon [16] и языке программирования Python [17].

Для хранения данных была выбрана СУБД PostgreSQL – это свободная объектно-реляционная система управления базами данных, идеально подходящая для UNIX-подобных систем, таких как: MacOS и Ubuntu.

Для тестирования работы серверной части следует использовать автоматизированную структуру Taurus.

2.2 Проектирование архитектуры программного обеспечения проекта

При разработке программного продукта использовалась двух уровневая архитектура «клиент-сервер», представленная на рисунке 17.

Рисунок 17 – Двухуровневая архитектура «клиент-сервер»

В двухуровневой модели база данных хранится на сервере. На сервере же находится ядро СУБД. На клиенте располагается презентационная логика и бизнес-логика приложения. Клиент обращается к серверу с запросами для получения данных [18].

В данной выпускной квалификационной работе в качестве клиента выступает разрабатываемое мобильное приложение для обучения элементам воднолыжного спорта «WakeboardTeacher», а в качестве сервера –

программное обеспечение, разрабатываемое с помощью языка программирования Python и средств фреймворка Falcon. Достоинством такой архитектуры является то, что все данные хранятся на серверной части приложения, что позволяет минимизировать программный код клиентского приложения и заполнять содержимое экрана динамически.

2.3 Проектирование интерфейса пользователя

На основе полученных функциональных требований был спроектирован пользовательский интерфейс мобильного приложения «WakeboardTeacher». Основными страницами приложения являются:

– страница описания тренера, проводящего обучение;
– страница уроков, отображающая список имеющихся материалов;
– страница выбранного урока;
– страница тестирования;
– страница, содержащая полезную информацию;
– страница соревнований, отображающая список предстоящих соревнований;
– страница выбранного соревнования;
– страница месторасположения вейк станций (вейкпарка);
– страница выбранной вейк станции (вейкпарка).

Рассмотрим элементы, расположенные на странице описания тренера, представленные на рисунке 18 и рисунке 19.

Страница описания тренера состоит из следующих компонентов: 1 – текстовое описание тренера; 2 – навигационная панель; 3 – ссылка на страницу социальной сети ВКонтакте; 4 – ссылка на youtube канал тренера.

Данная страница содержит текстовое поле (элемент 1), предназначенное для описания тренера, проводящего обучение, навигационную панель (элемент 2), позволяющую перемещаться между разделами приложения, а

также ссылки для посещения личной страницы тренера (компонент 3) в социальной сети и youtube канала (компонент 4).

Рисунок 18 – Страница описания тренера

Рисунок 19 – Страница описания тренера

На странице уроки, представленной на рисунке 20, размещены следующие: 1 – список видеоуроков; 2 – кликабельный элемент списка; 3 – навигационная панель.

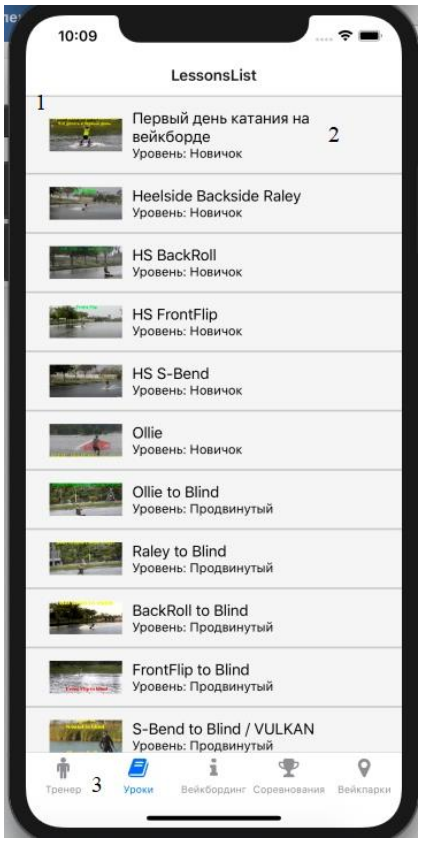


Рисунок 20 – Страница уроков

Данная страница состоит из списка видеоуроков (1 элемент), при нажатии на элемент (компонент номер 2) которого пользователь переходит на страницу конкретно выбранного урока, а также навигационную панель (элемент 3).

На рисунке 21 представлена страница выбранного урока, состоящая из следующих частей: 1 – видеоурок; 2 – текстовое описание основных моментов; 3 – кнопка, позволяющая пользователю скачать видеоматериал; 4 – навигационная панель; 5 – кнопка навигации; 6 – кнопка «Пройти тестирование».

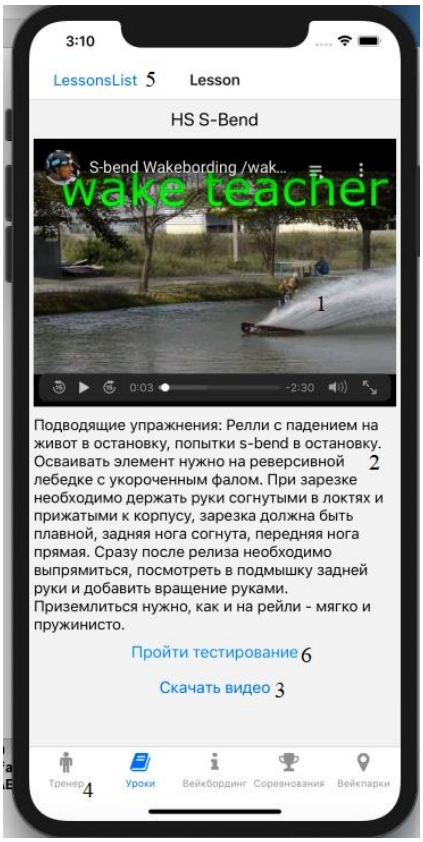


Рисунок 21 – Страница выбранного урока

Страница содержит 6 элементов. Первым элементом является видео урок, посвященный выбранному элементу. Под ним размещено краткое описание действий необходимых для правильного выполнения элемента. Элементом под номером 3 является кнопка, дающая возможность пользователю скачать видеоурок на мобильный телефон. Четвертый элемент – навигационная панель. Элемент номер 5 так же является элементом навигации, предназначенным для возврата пользователя к списку уроков. Шестым компонентом является кнопка, позволяющая пользователю проверить свои познания в технике выбранного элемента.

Страница тестирования представлена на рисунке 22 и предназначена для проведения опроса пользователя. Она содержит 4 компонента: 1 – список вопросов с тремя вариантами ответов; 2 – кнопка «Протестировать»; 3 – навигационная панель; 4 – кнопка навигации.

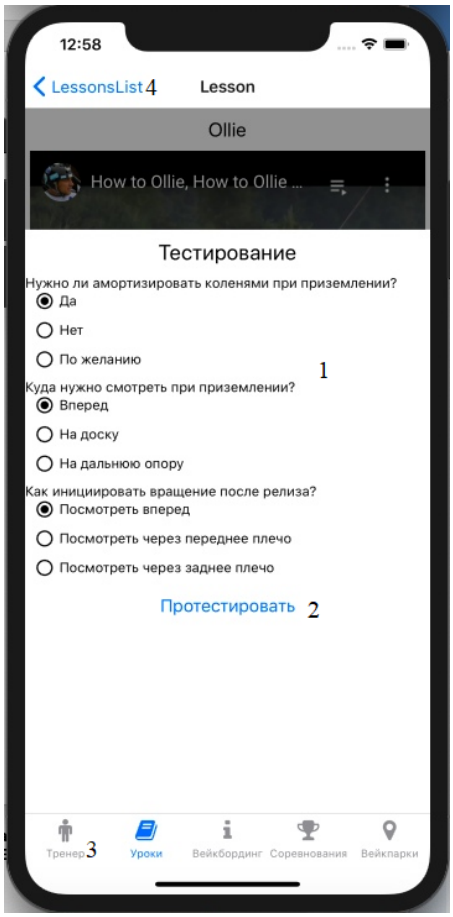


Рисунок 22 – Страница тестирования

Страница состоит из 4 компонентов. Первым элементом является список вопрос с вариантами ответа. Компонентом номер 2 является кнопка, при нажатии на которую происходит подсчет правильных ответов и выведение соответствующего текстового сообщения. Элемент 3 – навигационное меню. Компонент номер 4 – кнопка навигации, позволяющая вернуться к списку уроков.

Страница, содержащая полезную информацию, как для начинающего райдера, так и для совершенствующегося любителя вейкбординга представлена на рисунке 23. Она состоит из следующих компонентов: 1 – кнопка «История»; 2 – кнопка «Советы для начинающих»; 3 – кнопка «Техника безопасности»; 4 – кнопка «Выбор оборудования»; 5 – навигационная панель.

Рисунок 23 – Страница, отображающая информацию

Страница состоит из 5 частей. Элементы 1-4 являются кнопками. Первый компонент отображает информацию об истории возникновения данного вида спорта. Второй элемент содержит полезный советы для начинающих вейкбордистов. Данные советы должны помочь пользователю справиться с волнением перед первым стартом, а также подсказать что ему следует делать. Компонент номер 3 описывает общие пункты техники безопасности, которых придерживаются все вейк станции в мире. Элемент номер 4 поможет пользователю выбрать оборудование для катания. Данный компонент подробно расскажет обо всех аспектах выбора экипировки для занятия вейкбордингом. Компонент номер 5 – навигационное меню.

Страница соревнований представлена на рисунке 24 и состоит из следующих компонентов: 1 – список предстоящих соревнований; 2 – кликабельный элемент списка; 3 – навигационная панель.

Рисунок 24 – Страница соревнований

Страница состоит из 3 элементов. Первым компонентом является список всех предстоящих соревнований данного лета. В описании соревнования указана дата проведения, название состязания, а также место, где состоится борьба. Вторым компонентом является каждый элемент списка, при нажатии на который пользователь может подробно ознакомится с информацией о выбранном соревновании. Элемент номер 3 – навигационная панель.

Страница выбранного соревнования представлена на рисунке 25 и содержит 3 компонента: 1 – текстовая информация о соревновании; 2 – кнопка «Регистрация»; 3 – навигационная панель; 4 – кнопка навигации.

Страница состоит из 4 основных элементов. Первым компонентом является текстовая информация о соревновании: дата начала, место проведения, вейк парк, в котором состоится борьба, призовой фонд соревнований, стартовый взнос, который участнику необходимо внести для участия и контактный адрес, обратившись по которому можно уточнить все детали.

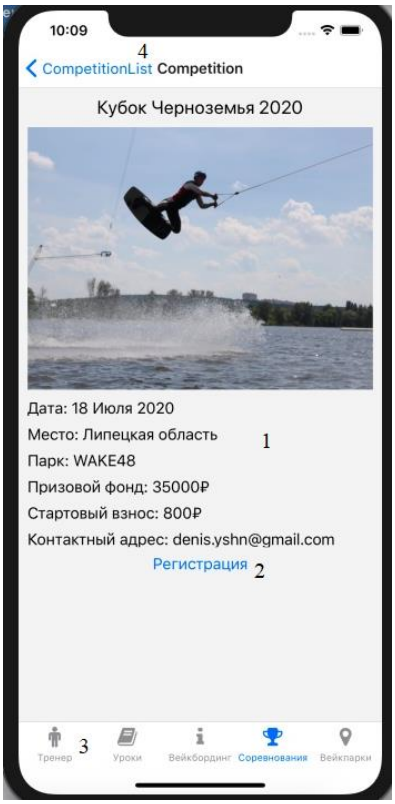


Рисунок 25 – Страница выбранного соревнования

На рисунке 26 представлена страница месторасположения всех близлежащих вейк станций.

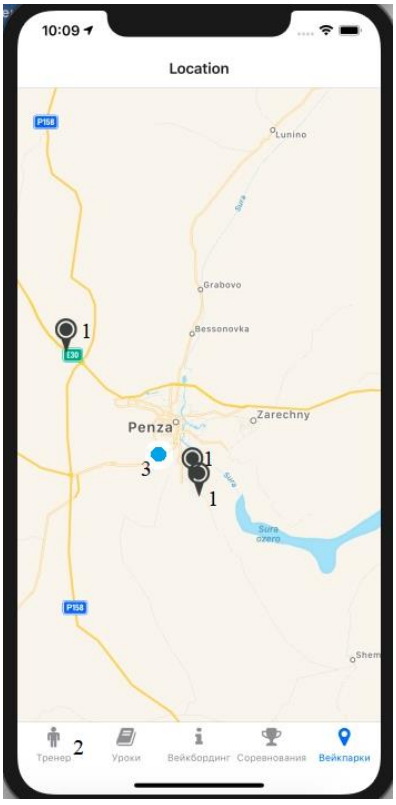


Рисунок 26 – Страница месторасположения всех станций

Страница месторасположения всех близлежащих вейк станций представлена на странице 26 и содержит карту, на которой маркерами отображены вейкпарки (элемент 1). При нажатии на маркер пользователь получит всю информацию о выбранной вейк станции. Компонент номер 2 – навигационная панель. Кроме того, на странице представлен элемент 3 – текущая местоположение пользователя.

Страница выбранной вейк станции представлена на рисунке 27 и содержит 3 компонента:1 – информацию о выбранной вейк станции; 2 – контактный телефон; 3 – навигационную панель.

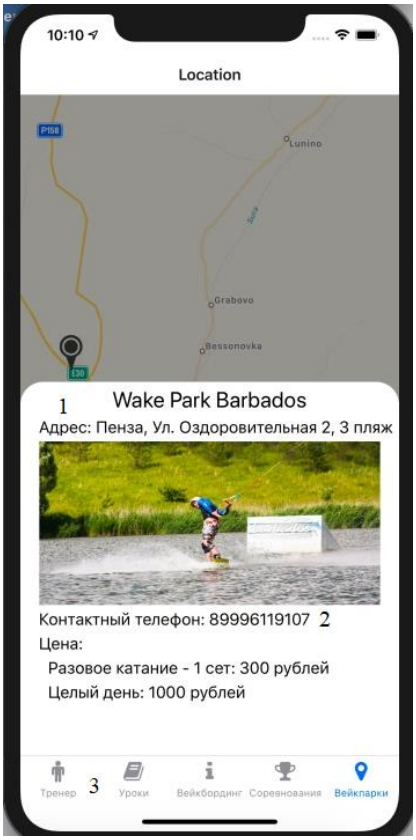


Рисунок 27 – Страница выбранной вейк станции

Элемент 1 представляет из себя текстовую информацию о вейк парке, которая сопровождается яркой цветной картинкой – изображением выбранной вейк станции. Компонент номер 2 является кнопкой, при нажатии на которую пользователю будет предложено позвонить по указанному номеру. Элемент номер 3 – навигационная панель.

2.4 Проектирование серверной части приложения

Для хранения данных была выбрана система управления базами данных PostgreSQL. Основными достоинствами данной СУБД является высокая производительность расширяемость и встроенная поддержка данных в формате JSON.

Для разработки серверной части приложения был выбран высокоуровневый язык программирования Python. Данный язык программирования позволяет работать быстро и осуществлять интеграцию систем более эффективно.

Для соединения базы данных и серверной части приложения используется Object Relational Mapping (ORM) ─ это технология, которая связывает базу данных с концепциями объектно-ориентированных языков программирования [19]. В данной выпускной квалификационной работе в качестве ORM используется peewee. Она поддерживает PostgreSQL и подходит для разработчиков Python.

В качестве среды разработки использовался редактор исходного кода, разработанный Microsoft – Visual Studio Code [20].

2.4.1 Логическое проектирование базы данных

На основе анализа предметной области и полученных функциональных требований была построена концептуальная модель данных. Элементами исходной концептуальной модели являются сущности и связи между ними предметной области «Обучение элементам воднолыжного спорта». Сущности моделируются исходя из состава выходных данных. Связи между сущностями показывают семантику взаимоотношения сущностей в предметной области информационной системы.

Ниже в таблице 7 представлены типы сущностей предметной области.

Таблица 7 – Типы сущностей

№	Наименование	Описание	Категория	Количество экземпляров	Режим обновления
1	Точки	Список маркеров, отображающих месторасположение вейк станций	Стержневая	1500	5% в год
2	Уроки	Список уроков	Стержневая	200	5% в год
3	Соревнования	Список соревнований	Стержневая	100	5% в год
4	Вопросы	Список вопросов к урокам	Стержневая	1000	5% в год

Были выделены следующие типы связи между типами сущностей предметной области, которые представлены в таблице 8.

Таблица 8 – Типы связей

№	**Тип сущности 1**	**Тип сущности 2**	**Описание**	**Мощность**	**Обязательность**	**Ограничения**
1	Уроки	Вопросы	После урока пользователь проходит тестирование	1 : M	Обязательно	

Описание атрибутов всех сущностей приведено в приложении Б.

Логическая модель базы данных представлена на рисунке 28.

Рисунок 28 – Логическая модель базы данных

Логическое проектирование отображает концептуальную модель предметной области в объекты используемой модели данных.

2.4.2 Проектирование API

Серверная часть приложения должна представлять из себя REST API. Во время разработки данного сервиса, необходимо сосредоточить внимание на ресурсах приложения. Для получения мобильным приложением данных с серверной части следует использовать стандартные методы HTTP. Для идентификации ресурса используется Uniform Resource Identifier (URI) – универсальный идентификатор ресурса. Для мобильного приложения следует создать 4 ресурса:

- списка уроков;
- списка соревнований;
- списка маркеров;
- списка вопрос.

Для работы с базой данных необходимо использовать Object Relational Mapping (ORM). Данная технология позволяет связывать базу данных с концепциями объектно-ориентированных языков программирования. Для реализации задуманного следует создать классы на языке программирования Python. Атрибуты разработанных классов будут представлять атрибуты базы данных. Также необходимо создать класс, определяющий URI к ресурсам, хранящимся в базе данных.

Для создания серверного приложения требуется использовать Falcon Web Framework – это фреймворк, включающий HTTP и архитектурный стиль REST. Falcon обладает высокой производительностью и интуитивно понятными объектами запроса и ответа. Также достоинством выбранного средства является поддержка языка Python. Кроме того, он используется такими компаниями как: EMC, Opera Software, Cronitor.

Таким образом были выделены средства для разработки серверной части разрабатываемого программного продукта.

3 Реализация мобильного приложения

3.1 Реализация серверной части

3.1.1 Реализация базы данных

Создание таблиц базы данных было выполнено запуском функции init_tables() в созданном ранее виртуальном окружении Python 3 для программного проекта. Для заполнения таблиц были разработаны функции generate_points(), generate_competitions(), generate_questions(), generate_lessons(). Аргументом данных функций является число, отображающее количество создаваемых строк. Заполнение строк актуальной информацией производилось в многофункциональной платформе для администрирования pgAdmin 4 [21].

Полученная физическая модель представлена на рисунке 29.

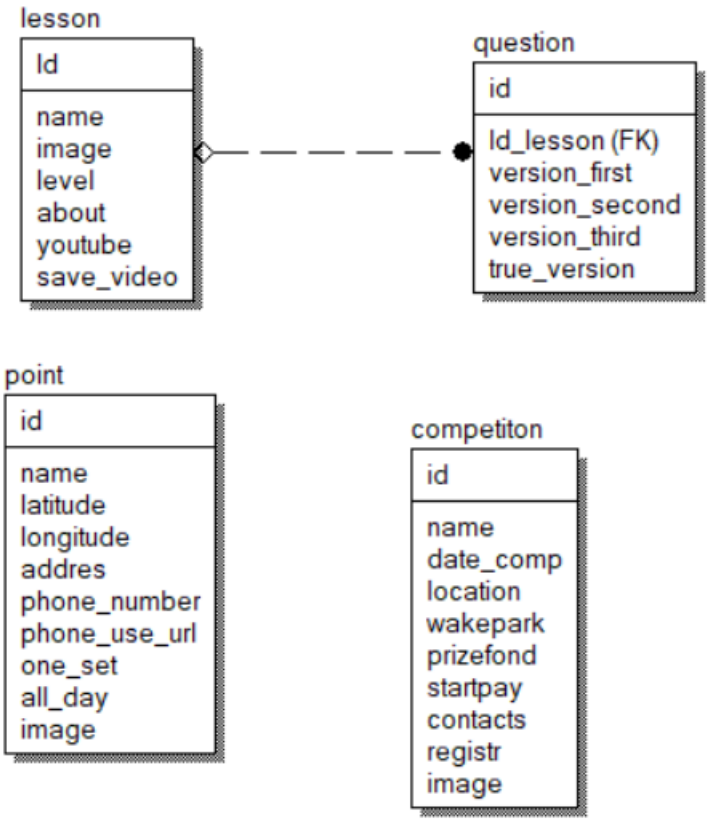


Рисунок 29 – Физическая модель базы данных

Исходный код функций заполнения таблиц базы данных представлен в классе models.py в приложении В.

3.1.2 Разработка серверной части приложения

Серверная часть программного продукта была реализована с помощью фреймворка Falcon, который служит для поддержки REST API, и языка программирования Python. В качестве ORM использовался Peewee ORM. Он поддерживает Postgresql и позволяет использовать его разработчикам Python.

Для соединения базы данных с мобильным приложением были разработаны несколько классов на языке Python.

Класс app.py реализует создание URI. Он содержит маршрутизацию для созданных ресурсов, основную она возможностях фреймворка Falcon.

Класс models.py содержит классы описывающие сущности созданной базы данных, а также класс подключения базы данных. Атрибуты классов содержат описание атрибутов базы данных.

Класс competition.py реализует запросы на получение коллекции списка соревнований и конкретного соревнования по указанному id.

Класс lesson.py осуществляет запросы, извлекающие коллекцию уроков и конкретного урока по указанному id.

Класс point.py создает запросы на получение коллекции списка маркеров и одного маркер по указанному id.

Класс question.py осуществляет запросы на получение коллекции списка вопросов по указанному id урока.

Серверная часть развернута на локальном сервер. Обращение к данным происходит с помощью HTTP запроса GET с указанием названия ресурса. Например, для получения списка уроков следует использовать URL https://localhost:8000/lesson, а для получения данных конкретного урока следует воспользоваться https://localhost:8000/lesson/2.

Исходный код созданных классов серверной части представлен в приложении В.

3.2 Реализация мобильного приложения

Пользовательский интерфейс кроссплатформенного мобильного приложения «WakeboardTeacher» был реализован с помощью стандартных средств XCode 11, фреймворка React Native и языка программирования JavaScript.

Экран приветствия создан с помощью xib – средства среды разработки XCode 11. Xib файлы позволяют создавать дизайн элементов и экранов используя библиотеку стандартных графических компонентов.

Помимо экрана приветствия приложение содержит еще несколько страниц:

- страница описания тренера;
- страница, содержащая список уроков;
- страница выбранного урока;
- страница тестирования;
- страница полезной информации о вейкбординге;
- страница, содержащая список соревнований;
- страница выбранного соревнования;
- страница карты маркерами, отображающими расположение вейк станций;
- страница выбранной вейк станции.

На странице описания тренера в верхней части экрана расположено название страницы. Под названием размещено фото тренера, его краткое описание, а также две ссылки: на личную страницу Владимира в социальной сети VKontakte, что позволит пользователю задать ему вопрос, и на YouTube канал WakeboardTeacher. В нижней части экрана находиться навигационная панель, которая дает пользователю возможность перемещаться между страницами приложения.

Второй вкладкой навигационной панели является страница, содержащая список уроков. Она состоит из названия, списка уроков, элементы которого

включают изображение, название урока и уровень катания. Все элементы списка являются кликабельными, что позволяет перейти на страницу выбранного урока при нажатии. В нижней части страницы располагается навигационное меню.

Страница выбранного урока состоит из названия урока, обучающего видео, краткого описания техники исполнения элемента, а также ссылки на веб-сервис, позволяющий пользователю скачать видео на мобильное устройство, и кнопки «Пройти тестирование». В нижней части экрана находиться навигационный список кнопок.

На странице тестирования расположен список вопрос, вариантов ответов и кнопка «Протестировать». По результатам опроса приложение выдает соответствующее сообщение на экран. Внизу экрана также располагается навигационная панель.

Третьей вкладкой навигационного меню является страница полезной информации о вейкбординге. Она содержит название, список кликабельных элементов, при нажатии на которые происходит отображение интересующей информации, и навигационную панель для перемещения между экранами приложения.

Следующей вкладкой навигации является страница, содержащая список предстоящих соревнований. Она состоит из списка, содержащих кликабельные элементы, хранящие информацию о дате проведения, названии соревнования и месте их проведения, при нажатии на которые пользователь переходит на страницу выбранного соревнования. В нижней части экрана располагается навигационная панель.

Страница выбранного соревнования содержит название соревнования, фотографию, информацию о дате, месте проведения, сумме денежного взноса, контактные данные, а также кнопку «Регистрация», позволяющая пользователю перейти на сайт wakebase.ru, где пользователь может зарегистрироваться на участие в данном мероприятии.

Последней вкладкой навигационного меню является страница карты. Она представляет из себя карту, с указанными на ней маркерами – вейк станциями. Кроме того, она содержит маркер с текущим месторасположением пользователя и навигационную панель в нижней части экрана.

При нажатии на маркер пользователь переходит на страницу, отражающим информацию о выбранной вейк станции: название, изображение, контактный номер, при нажатии на который мобильное приложение переходит в приложение «Телефон» и вызывает по указанному номеру, а также стоимость катания: за разовое катание и за целый день.

Все указанные выше страницы приложения реализованы с помощью средств фреймворка React Native и языка программирования JavaScript.

Для реализации навигации использовались маршрутизация и навигация для react native приложений. Схема маршрутизации приложения приведена на рисунке 30.

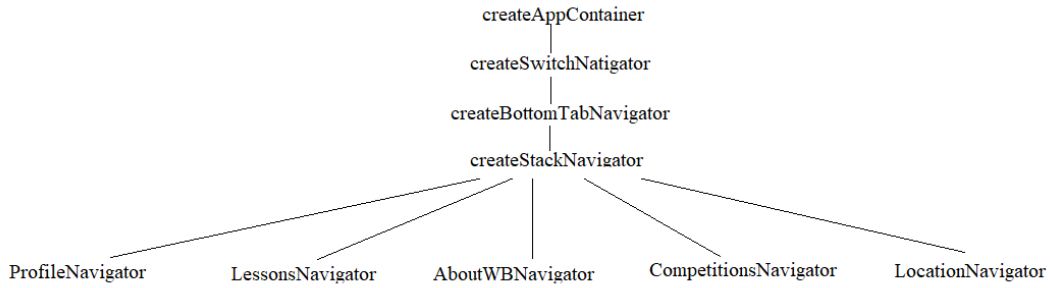


Рисунок 30 – Маршрутизация мобильного приложения

Для создания навигационной панели использовался маршрутизатор createBottomTabNavigator. Для реализации навигации между экранами приложения использовался маршрутизатор createStackNavigator [22].

Ниже приведена спецификация классов программного продукта.

AppNav.js , Index.js – классы навигации.

App.js – класс макета приложения.

Api.js – класс, реализующий метод сопряжения мобильного приложения с серверной частью программного продукта.

Все представленные ниже классы наследуются от стандартного класса React Native Component и реализуют логику отображение компонентов экрана.

ProfileScreen.js – класс экрана описания тренера.

ListScreen.js – класс экрана списка уроков.

LessonScreen.js – класс экрана выбранного урока.

Test.js – класс экрана тестирования.

AboutWBScreen.js – класс экрана, содержащего полезную информацию о вейкбординге.

CompetitionListScreen.js – класс экрана списка соревнований.

CompetitionScreen.js – класс экрана выбранного соревнования.

LocationParksScreen.js – класс экрана расположения вейк станций.

SideBarPark.js – класс экрана описания выбранной вейк станции.

На рисунке 31 представлена диаграмма классов мобильного приложения.

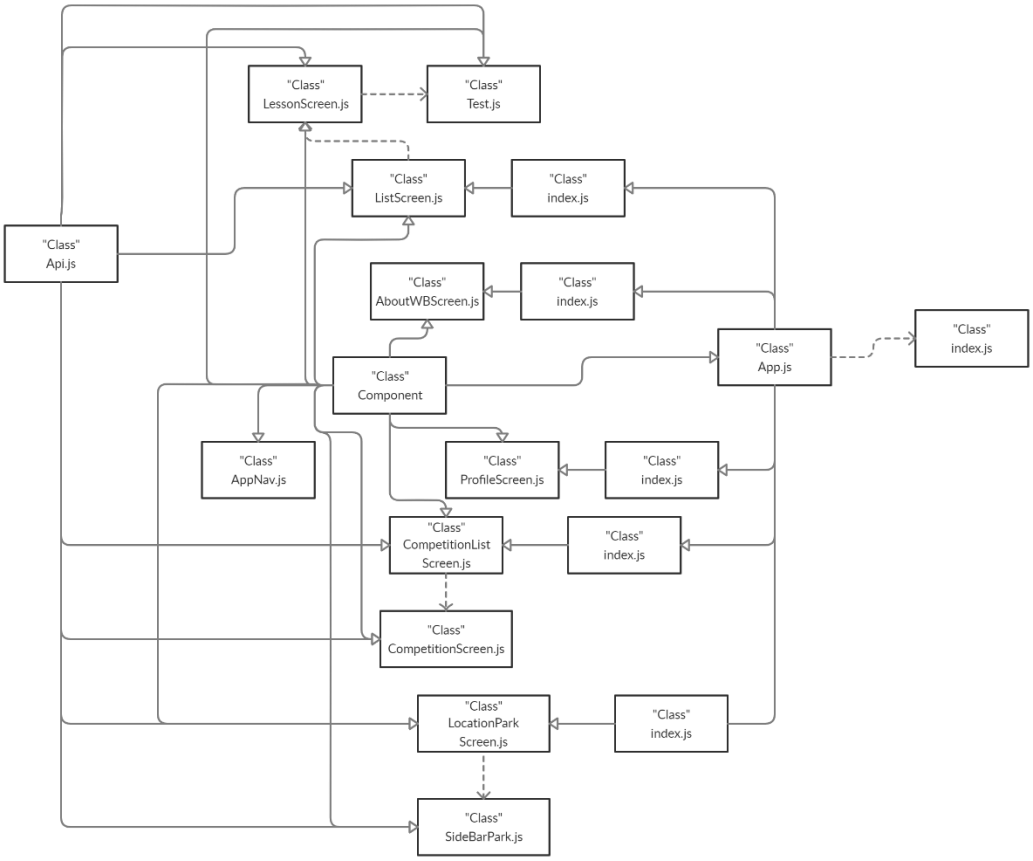


Рисунок 31 – Диаграмма классов мобильного приложения

Исходный код мобильного приложения представлен в приложении Г.

Для визуализации компонентов программы, существующих на этапе развертывания, используем диаграмму развертывания. На рисунке 32 представлена диаграмма развертывания мобильного приложения на IOS.

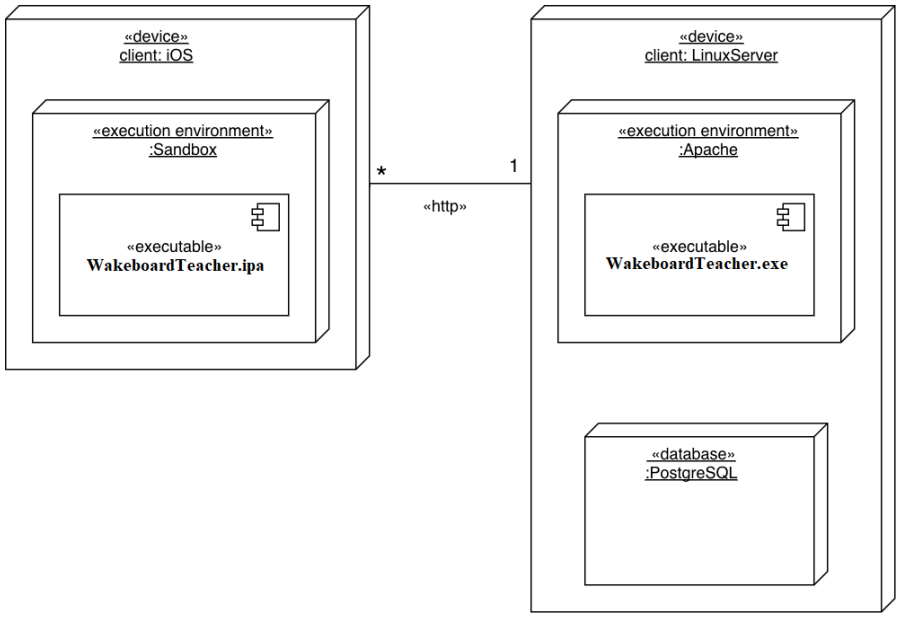


Рисунок 32 – Диаграмма развертывания мобильного приложения на IOS

На рисунке 33 изображена диаграмма развертывания мобильного приложения на Android.

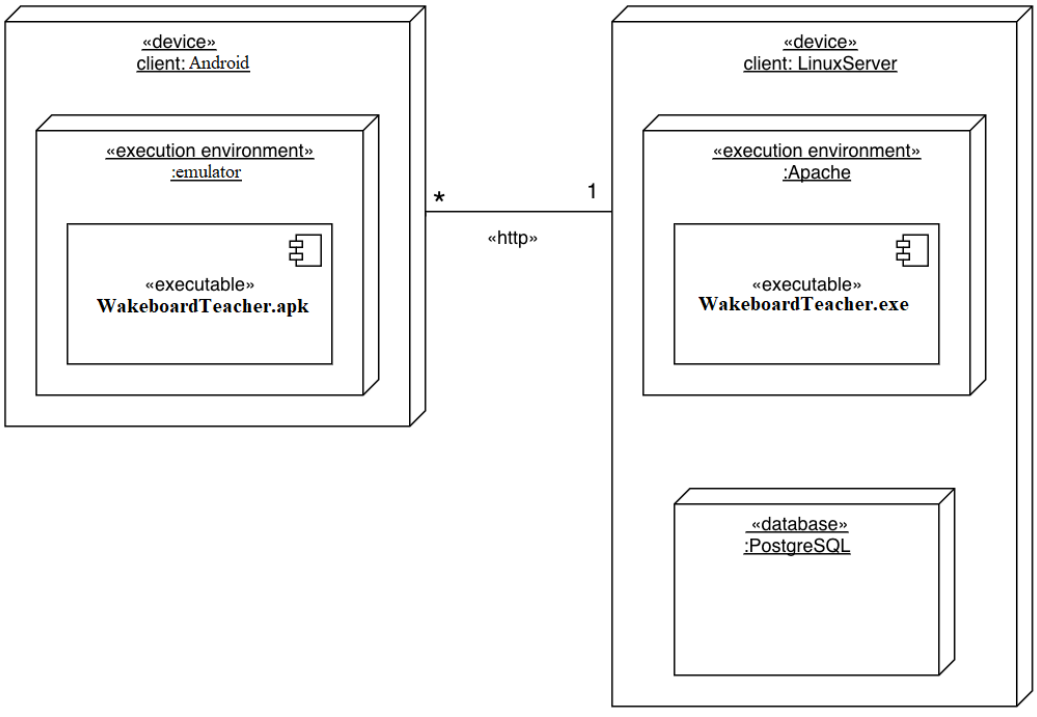


Рисунок 33 – Диаграмма развертывания мобильного приложения на Android

На рисунке 34 представлена диаграмма компонентов, показывающая общую структуру исходного кода программного кода.

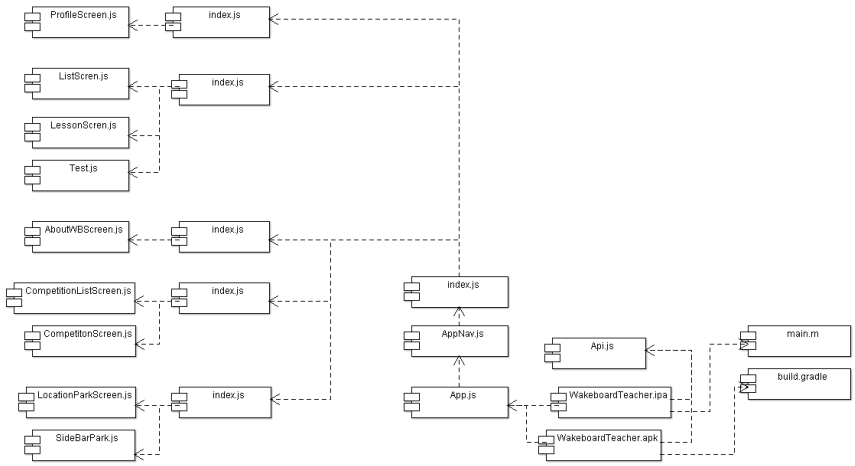


Рисунок 34 – Диаграмма компонентов исходного

Серверная часть приложения разворачивается на устройстве, работающем на Unix-подобной операционной системе. Также на нем запускается сервер базы данных. Клиентское приложение разворачивается на мобильных устройствах пользователя под управлением iOS или Android и взаимодействуют с сервером с помощью протокола HTTP.

4 Контроль качества программного продукта

4.1 Тестирование серверной части

Для проверки работоспособности серверной части разработанного продукта используем браузер «Chrome». Адресную строку браузера будем заполнять ссылками запросов. Ответом должна быть JSON строка с запрашиваемыми данными. Запросы приведены ниже в таблице 9. Скриншоты результатов запросов представлены в приложении Д.

Таблица 9 – Результаты функционального тестирования приложения

Номер теста	Название ресурса	Описание теста	Запрос		Результат
			Метод	URI	
1	lesson	Получение списка уроков	GET	/lessons	Тест пройден. Рисунок Д.1
2		Получение данных урока номер 2	GET	/lessons/2	Тест пройден. Рисунок Д.2
3	point	Получение списка маркеров	GET	/points	Тест пройден. Рисунок Д.3
4		Получение данных маркера номер 1	GET	/points/1	Тест пройден. Рисунок Д.4
5	competiton	Получение списка соревнованиий	GET	/competitions	Тест пройден. Рисунок Д.5
6		Получение данных о соревнованиях номер 5	GET	/competitions/5	Тест пройден. Рисунок Д.6.
7	question	Получение списка вопросов	GET	/questions	Тест пройден. Рисунок Д.7
8		Получение списка вопросов относящихся к 7 уроку	GET	/questions?lesson_id=2	Тест пройден. Рисунок Д.8

Для оценки производительности нашего API воспользуемся автоматизированной структурой Taurus [23]. Для этого необходимо установить структуру в наше виртуальное окружение и создать файл конфигурации теста. Этот тест симулирует веб трафик 100 пользователей, с нарастанием их числа в течении одной минуты, и держит нагрузку 2 минуты 30 секунд. Затем следует запустить сервер с указанием количества рабочих процессов. В нашем случае этим числом является 1. После этого запускаем

Taurus. В время первого запуска происходит загрузка и установка всех зависимостей, после чего открывается консоль с ходом исполнения теста. Она представлена на рисунке 35.

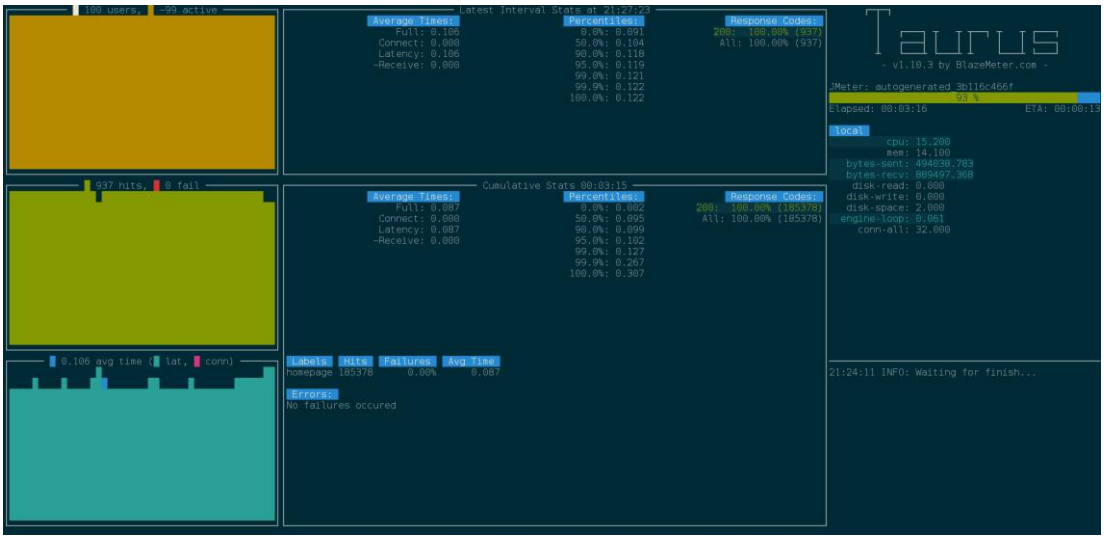


Рисунок 35 – Консоль структуры «Taurus»

Для качественного отображения результата используем опцию –report при запуске «Taurus», что позволит сгенерировать веб-отчет. Результаты веб-отчета нагрузочного тестирования представлены на рисунке 36.

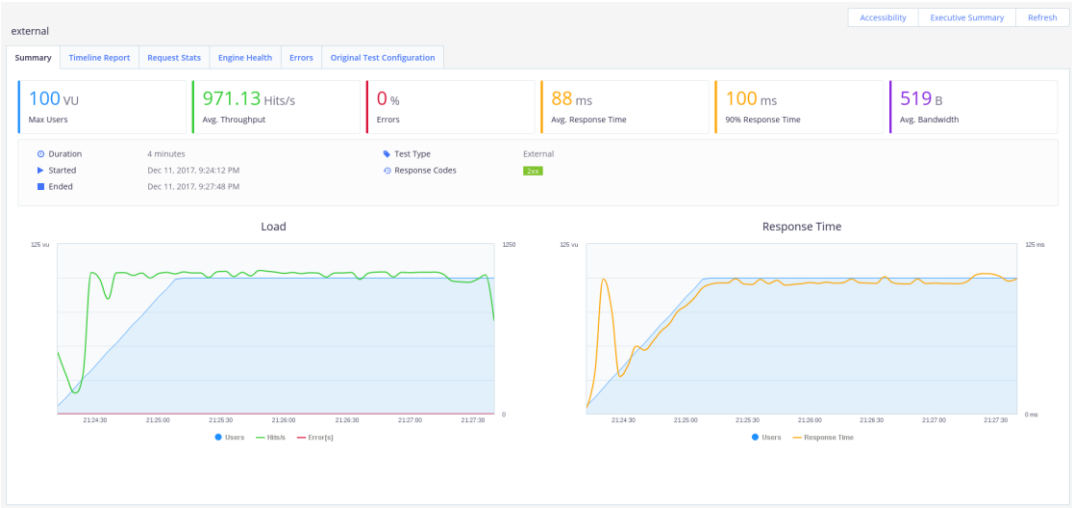


Рисунок 36 – Веб-отчет по работе API

По результатам проведенного нагрузочного тестирования, можно сделать вывод, что реализованное API прекрасно справляется с количеством пользователей равным 100. Пропускная способность близка к 1000 запросов в секунду, ошибки полностью отсутствуют, а среднее время ответа равно десятой доли секунды.

Все тесты API пройдены успешно, что свидетельствует о корректной работе серверной части программного продукта.

4.2 Тестирование клиентской части приложения

4.2.1 Проверка эргономичности пользовательского интерфейса

Тестирование программного обеспечения – процесс исследования, испытания программного продукта, имеющий своей целью проверку соответствия между реальным поведением программы и ее ожидаемым поведением на конечном наборе тестов, выбранных определенным образом.

Проведем юзабилити-тестирование для проверки удобства интерфейса пользователя на мобильных устройствах с операционными системами Android и IOS [24]. Для этого были привлечены представители целевой аудитории, перед которыми были поставлены задачи, представленные в таблице 10. Стоит также отметить, что ожидаемым результатом выполнения каждой задачи является беспрепятственное обнаружение компонентов приложения. Скриншоты результатов тестировая представлены в приложении Е.

Таблица 10 – Результаты юзабилити-тестирования

Номер задачи	**Выполняемые действия**	**Полученный результат**	**Полученный результат на Android**	**Полученный результат на IOS**
1	Поиск страницы «тренер»	Все участники тестирования справились с поставленной задачей незамедлительно.	Рисунок Е.1а, Рисунок Е.1б	Рисунок Е.2а, Рисунок Е.2б
2	Поиск страницы «уроки»	Все участники тестирования справились с поставленной задачей незамедлительно.	Рисунок Е.3	Рисунок Е.4
3	Поиск урока «TS Backroll to Revert»	Все участники тестирования справились с поставленной задачей незамедлительно.	Рисунок Е.5	Рисунок Е.6

Продолжение таблицы 10

4	Поиск кнопки «Пройти тестирование»	Большая часть участников тестирования справились с поставленной задачей.	Рисунок Е.7	Рисунок Е.8
5	Поиск страницы «Вейкбординг»	Все участники тестирования справились с поставленной задачей незамедлительно.	Рисунок Е.9	Рисунок Е.10
6	Поиск информации о истории создания вейкбординга	Все участники тестирования справились с поставленной задачей незамедлительно.	Рисунок Е.11	Рисунок Е.12
7	Поиск советов для начинающих	Все участники тестирования справились с поставленной задачей незамедлительно.	Рисунок Е.13	Рисунок Е.14
8	Поиск раздела «Техника безопасности»	Все участники тестирования справились с поставленной задачей незамедлительно.	Рисунок Е.15	Рисунок Е.16
9	Поиск раздела «Выбор оборудования»	Все участники тестирования справились с поставленной задачей незамедлительно.	Рисунок Е.17	Рисунок Е.18
10	Поиск страницы «Соревнования»	Все участники тестирования справились с поставленной задачей незамедлительно.	Рисунок Е.19	Рисунок Е.20
11	Поиск соревнования «Кубок Черноземья 2020»	Все участники тестирования справились с поставленной задачей незамедлительно.	Рисунок Е.21	Рисунок Е.22
12	Поиск страницы «Вейкпарки»	Все участники тестирования справились с поставленной задачей незамедлительно.	Рисунок Е.23	Рисунок Е.24

Окончание таблицы 10

13	Поиск вейк станции «Wake Park Sura»	У половины участников тестирования вознили трудности.	Рисунок Е.25	Рисунок Е.26

По результатам проведенного тестирования можно сделать вывод, что большая часть страниц мобильного приложения на операционных системах Android и IOS являются эргономичными и удобными для пользователей. Созданная система разделов и навигации приложения существенно упростила представителям целевой аудитории задачу прохождения тестирования. Однако, при проведении исследования у половины участников возникли сложности с поиском вейк станции «Wake Park Sura». Это связано с отсутствием поясняющих записей рядом с маркерами на карте. Выявленный недочет будет исправлен в следующей версии приложения.

4.2.2 Проверка функциональных возможностей приложения

Проведем функциональное тестирование для проверки корректности работы выделенных ранее функциональных требований. Результаты представлены в таблице 11. Скриншоты результатов приведены в приложении Ж.

Таблица 11 – Результаты тестирования

Номер теста	**Функциональное требование**	**Выполняемые действия**	**Ожидаемый результат**	**Полученный результат на Android**	**Полученный результат на IOS**
1	Посмотреть информации о тренере	Перейти на страницу «тренер»	Отображение всех компонентов экрана	Рисунок Ж.1а, Рисунок Ж.1б	Рисунок Ж.2а, Рисунок Ж.2б
2	Задать тренеру вопрос	Перейти по ссылке «Задать вопрос тренеру»	Открытие страницы Владимира в социальной сети ВК	Рисунок Ж.3	Рисунок Ж.4

Продолжение таблицы 11

3	Посмотреть youtube канал тренера	Перейти по ссылке «YouTube канал Wakeboard Teacher»	Открытие приложения YouTube на канале «Wakeboard Teacher»	Рисунок Ж.5	Рисунок Ж.6
4	Просмотреть список видеоуроков	Перейти на страницу «уроки»	Отображение списка уроков	Рисунок Ж.7	Рисунок Ж.8
5	Посмотреть видеоурок	Выбрать элемент списка «TS Backroll to Revert»	Отображение компонентов страницы урока	Рисунок Ж.9	Рисунок Ж.10
6	Скачать видеоурок на мобильное устройство	Нажать кнопку «Пройти тестирование»	Открытие веб-сервиса с выбранным уроком	Рисунок Ж.11	Рисунок Ж.12
7	Пройти тестирование	Перейти по ссылке «скачать видео»	Отображение списка вопросов с вариантами ответов	Рисунок Ж.13	Рисунок Ж.14
8	Посмотреть полезную информацию	Перейти на страницу «Вейкбординг»	Отображение компонентов списка	Рисунок Ж.15	Рисунок Ж.16
9	Прочитать историю	Выбрать элемент «История»	Отображение информации о истории создания вейкбординга	Рисунок Ж.17	Рисунок Ж.18
10	Прочитать советы для начинающих	Выбрать элемент «Советы для начинающих»	Отображение советов для начинающих	Рисунок Ж.19	Рисунок Ж.20
11	Ознакомиться с техникой безопасности	Выбрать элемент «Техника безопасности»	Отображение техники безопасности	Рисунок Ж.21	Рисунок Ж.22
12	Узнать о выборе оборудования	Выбрать элемент «Выбор оборудования»	Отображение информации о выборе оборудования	Рисунок Ж.23	Рисунок Ж.24
13	Просмотреть список соревнований	Перейти на страницу «Соревнования»	Отображение списка соревнований	Рисунок Ж.25	Рисунок Ж.26
14	Посмотреть информацию о предстоящем соревновании	Выбрать элемент списка «Кубок Черноземья 2020»	Отображения данных о соревнованиях	Рисунок Ж.27	Рисунок Ж.28

Окончание таблицы 11

15	Зарегистрироваться в соревнованиях	Перейти по ссылке «Регистрация» и заполнить поля	Открытие страницы регистрации на сайте wakebase.ru	Рисунок Ж.29	Рисунок Ж.30
16	Посмотреть месторасположение ближайших вейк станций	Перейти на страницу «Вейкпарки»	Отображение карты и маркеров	Рисунок Ж.31	Рисунок Ж.32
17	Посмотреть информацию о выбранной вейк станции	Нажать на маркер	Отображение информации о вейк станции	Рисунок Ж.33	Рисунок Ж.34
18	Позвонить по контактному номеру	Нажать на «Контактный телефон»	Переход в приложение «Телефон»	Рисунок Ж.35	Рисунок Ж.36

По результатам проведенного функционального тестирования можно сделать вывод, что все выделенные функциональные требования выполнены и работают в соответствии с ожидаемым результатом.

Следует также отметить, что некоторые возможности реального мобильного устройства недоступны на эмуляторе. Одним из таких является вызов номера в эмуляторе «Iphone 11 Max Pro». Средства фреймворка React Native указывают о невозможности вызова абонента из эмулятора.

4.2.3 Метрики кода

Приведенный в листинге 1 класс предназначен для получения данных с серверной части приложения.

Листинг 1 – Класс для работы с серверной частью приложения

```
import axios from 'axios';
const API_HOST = "https://private-51ec2-wakeboardteacherlessons.apiary-mock.com";
export default class Api {
  //метод создания заголовков запросов
  static async getHeaders(method) {
    const contentType = { 'Content-Type': 'application/json' };
    return {
      ...contentType
```

```
    }
  }
  //метод запроса get
  static get(url, config) {
    return this.doRequest(url, config, 'get');
  }
  //Метод, выполняющий запрос к серверной части приложения
  static async doRequest(url, conf, method) {
    const config = Object.assign({
      headers: await this.getHeaders(method),
      method: method,
      baseURL: API_HOST,
      url: url
    }, conf);
    return axios(config);
  }
}
```

Представленный в Листенге 1 класс используем для вычисления метрик кода.

4.2.3.1 Метрика Холстеда

Входными данными для метрик Холстеда являются уникальные операторы и операнды. В таблице 12 представлены уникальные операторы.

Таблица 12 – Уникальные операторы

Номер	Оператор
1	=
2	.
3	:

В таблице 13 представлены уникальные операнды.

Таблица 13 – Уникальные операнды

Номер	Оператор
1	API_HOST
2	contentType
3	config
4	headers
5	method
6	baseURL
7	url
8	conf

Метрики Холстеда основаны на следующих показателях:

– n1 – число уникальных операторов программы, включая символы-разделители, имена процедур и знаки операций (словарь операторов);

– n2 – число уникальных операндов программы (словарь операндов);

– N1 – общее число операторов в программе;

– N2 – общее число операндов в программе;

– n1' – теоретическое число уникальных операторов;

– n2' – теоретическое число уникальных операндов.

Учитывая введенные обозначения, можно определить словарь программы по формуле:

$$n = n1 + n2.$$

Длина программы рассчитывается по формуле:

$$N = N1 + N2.$$

Теоретический словарь программы:

$$n' = n1' + n2'.$$

Теоретическая длина программы (для стилистически корректных программ отклонение N от N' не превышает 10%):

$$N' = n1 * log_2 n1 + \ n2 * log_2 n2.$$

Объем программы:

$$V = N * log_2 n.$$

Уровень качества программирования, основанный лишь на параметрах реальной программы без учета теоретических параметров:

$$L' = \frac{2 * n2}{n1 * N2}.$$

Сложность понимания программы:

$$E_c = \frac{V}{L' * 2}.$$

Результаты расчета перечисленных метрик Холстеда представлены в таблице 14.

Таблица 14 – Расчет метрик Холстеда

Метрика	**Значение**
n1	3
n2	8
N1	14
N2	21
n	11
N	35
V	121.079
N'	28.755
L'	0.254
E_C	238.34

Метрика Ec отображает сложность понимания программы. Полученные результаты сложности показывают, что разработанный код имеет достаточно высокий уровень качества, что позволит легко поддерживать и изменять его при поддержке проекта.

4.2.3.2 Метрика уровня комментированности программы

Метрика отражает насыщенность программы комментариями. Оценка уровня комментированности программы F:

$$F = \frac{N_{\text{ком}}}{N_{\text{стр}}},$$

где $N_{\text{ком}}$ – количество комментариев,

$N_{\text{стр}}$ – количество строк или операторов.

Параметр F принято считать большим или равным 0.1. Это означает, что на каждые 10 строк программного кода должен быть минимум один. Однако, как правило, комментарии в коде расположены неравномерно: существенно преобладают в начале, а в середине или конце – недостаток. Поэтому приведенная выше формула отражает уровень комментированности не точно. Необходимо разбить программу на n равных сегментов и для каждого определять F_i:

$$F_i = sign\left(\frac{N_{\text{ком}}}{N_{\text{стр}}}\right).$$

Приведенный в листенге 1 класс разобьём на 3 сегмента по ~8 строк и для каждого определим F_i:

$$F_1 = (\frac{1}{8})=0.125,$$

$$F_2 = (\frac{2}{8})=0.25,$$

$$F_1 = (\frac{0}{9})=0,$$

$$F = \sum_{i=1}^{3} F_i = 0.375.$$

На основе представленных вычислений можно сделать вывод, что код класс обладает нормальным уровнем комментированности, так как F ≥ 0.1.

Заключение

В результате выполнения выпускной квалификационной работы были разработаны кроссплатформенное мобильное приложение, предоставляющее пользователю возможности по освоению элементов воднолыжного спорта «Вейкбординг», и серверное приложение, созданное для сопряжения мобильного приложения и базы данных, размещенной на серверной части программного продукта.

На этапе анализа предметной области были выявлены функциональные и нефункциональные требования к разрабатываемому программному продукту, а также рассмотрены аналогичные приложения, представленные на рынке.

На этапе проектирования была спроектирована логическая модель базы данных, серверная часть приложения, а также – интерфейс пользователя.

Мобильное приложение было разработано с помощью средств фреймворка React Native и языка программирования Javascript в среде разработке Visual Studio Code. Серверная часть программного продукта была реализована с помощью фреймворка Falcon и языка программирования Python в среде разработки Visual Studio Code.

Для проверки корректности работы программного продукта было проведено функциональное тестирование мобильного приложения и серверной части. Также были рассчитаны метрики кода, которые показали достаточно хорошее качество кода

В дальнейшем планируется развернуть серверную часть на облачном сервисе Heroku и опубликовать мобильное приложение на IOS в магазине приложений для iOS AppStore и мобильное приложение на Android в магазине приложение GooglePlay.

Список использованных источников

1. Программа «Цифровая экономика Российской Федерации». Утверждена распоряжением Правительства Российской Федерации от 28 июля 2017 г. № 1632-р. [Электронный ресурс]. – URL: http://static.government.ru/media/files/9gFM4FHj4PsB79I5v7yLVuPgu4bvR7M0.pdf (дата обращения: 10.05.2019).

2. Распоряжение Президента Российской Федерации от 9 мая 2017 г. № 203 «Стратегия развития информационного общества в Российской Федерации на 2017 – 2030 годы» [Электронный ресурс]. – URL: http://zmedu.hostedu.ru/files/ykaz_7668.pdf (дата обращения: 10.05.2020).

3. Краткое руководство по вейкбордингу. [Электронный ресурс]. – URL: https://www.tutorialspoint.com/wakeboarding/wakeboarding_quick_guide.htm (дата обращения 11.05.2020)

4. Влияние офисной работы на здоровье человека. [Электронный ресурс]. – URL: https://republic.ru/posts/92115 (дата обращения: 13.05.2020)

5. Спортивные буксировщики. [Электронный ресурс]. – URL: https://itboat.com/catalog/49-sportivnye-buksirovsiki (дата обращения: 15.05.2020)

6. Руководство для начинающих спортсменов. [Электронный ресурс]. – URL: https://www.yhwatersports.com/beginners-guide-2-wakeboarding/ (дата обращения: 16.05.2020)

7. Мобильное приложение «Вейкбординг». [Электронный ресурс]. – URL: https://apkpure.com/ru/%D0%B2%D0%B5%D0%B9%D0%BA%D0%B1%D0%BE%D1%80%D0%B4%D0%B8%D0%BD%D0%B3/wake.sb.st (дата обращения: 17.05.2020)

8. Мобильное приложение «Learn Wakeboarding Free». [Электронный ресурс]. – URL: https://apkpure.com/ru/learn-wakeboarding-free/com.tutorial.Wakeboarding (дата обращения: 17.05.2020)

9. WakeTips application. [Электронный ресурс]. – URL: https://www.wake.sg/waketips/ (дата обращения: 17.05.2020)

10. WebCanape. [Электронный ресурс]. – URL: https://www.web-canape.ru/business/vsya-statistika-interneta-na-2019-god-v-mire-i-v-rossii/ (дата обращения: 18.05.2020)

11. Estmax. [Электронный ресурс]. – URL: https://www.estmax.com/summer/veikbording-eto.html (дата обращения: 19.05.2020)

12. itBoat. [Электронный ресурс]. URL – https://itboat.com/news/309-mok-vklyuchil-veykbord-v-short-list-vidov-sporta-letney-olimpiady (дата обращения: 20.05.2020)

13. Методологии разработки программного обеспечения. [Электронный ресурс]. – URL: https://compress.ru/article.aspx?id=9633 (дата обращения: 20.05.2020)

14. UML – диаграммы классов. [Электронный ресурс]. – URL: https://prog-cpp.ru/uml-classes/ (дата обращения:21.05.2020)

15. React Native Documentation. [Электронный ресурс]. – URL: https://reactnative.dev/docs/ (дата обращения: 22.05.2020)

16. Framework Falcon Documentation. [Электронный ресурс]. – URL: https://falcon.readthedocs.io/en/stable/user/index.html (дата обращения: 24.05.2020)

17. Python Documentation. [Электронный ресурс]. – URL: https://www.python.org/doc/ (дата обращения: 26.05.2020)

18. Архитектура обработки данных [Электронный ресурс] Режим доступа: - https://studfile.net/preview/5265892/page:3/ (дата обращения 27.05.2020)

19. Peewee ORM манипуляция базами данных [Электронный ресурс]. – URL: https://python-scripts.com/peewee (дата обращения: 28.05.2020)

20. Visual Studio Code Documentation. [Электронный ресурс]. – URL: https://code.visualstudio.com/docs (дата обращения: 30.05.2020)

21. PgAdmin 4 Documentation [Электронный ресурс]. – URL: https://www.pgadmin.org/docs/pgadmin4/4.17/index.html (дата обращения: 31.05.2020)

22. React Navigation Documentation. [Электронный ресурс]. – URL: https://reactnavigation.org/docs/tab-based-navigation (дата обращения: 2.06.2020)

23. Taurus Doc. [Электронный ресурс]. – URL: https://gettaurus.org/ (дата обращения: 4.06.2020)

24. Юзабилити-тестирование [Электронный ресурс]. – URL: https://www.ashmanov.com/education/articles/yuzabiliti-testirovanie/ (дата обращения: 6.06.2020)

25. AppLine [Электронный ресурс]. – URL: https://www.appline.ru/services/testing/functionalnoe-testirovanie (дата обращения: 7.06.2020)

Приложение А.

ГЛОССАРИЙ

(справочное)

Android – мобильная операционная система для смартфонов, электронных планшетов и других устройств, разрабатываемая и выпускаемая американской компанией Google.

API (Application Programming Interface) – набор готовых классов, функций, процедур, констант и структур, предоставляемых приложением или операционной системой для использования во внешних программных продуктах.

IOS – мобильная операционная система для смартфонов, электронных планшетов и других устройств, разрабатываемая и выпускаемая американской компанией Apple.

ORM (Object Relational Mapping) ─ это технология, связывающая базу данных с концепциями объектно-ориентированных языков программирования.

RUP (Rational Unified Process) – методология разработки программного обеспечения, которая использует итеративную модель разработки.

HTTP (HyperText Transfer Protocol) – протокол прикладного уровня передачи данных, основанный на технологии «клиент-сервер».

UML (Unified Modeling Language) – язык графического описания для объектного моделирования в области разработки программного обеспечения, моделирования бизнес-процессов, системного проектирования и отображения организационных структур.

Фреймворк – это некий набор библиотек, который облегчает разработку программных продуктов продуктов, например, мобильных или десктопных приложений.

МОК (Международный Олимпийский Комитет) – международная организация, созданная для возрождения Олимпийских игр и пропаганды олимпийского движения.

Приложение Б

ОПИСАНИЕ АТРИБУТОВ СУЩНОСТЕЙ БАЗЫ ДАННЫХ

(рекомендуемое)

Таблица Б.1 – Атрибуты сущности «lesson»

Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
Id	Номер урока	Целый	4	0001-9999	нет	PK	1
name	Название урока	Символьный	255	‘А-я’, ‘ ’, ‘A-z’	нет		HS Raley
image	Ссылка на изображение	Символьный	255	‘А-я’, ‘ ’, ‘A-z’	нет		https://vk.com/away.php?to=http%3A%2F%2Fdrive.google.com%2Fuc%3Fexport%3Dview%26id%3D1dIawAa9VCof-bvZTXXUU1R8csWo1mUJA&cc_key=
level	Уровень урока	Символьный	255	‘А-я’, ‘ ’, ‘A-z’	нет		Уровень: Новичок
about	Описание урока	Символьный	255	‘А-я’, ‘ ’, ‘A-z’	нет		Подводящие упражнения: Олли на тяге 5-10 раз, Релли с падением на живот в остановку. Осваивать элемент нужно на реверсивной лебедке с укороченным фалом. При зарезке необходимо держать руки согнутыми в локтях и прижатыми к корпусу, зарезка должна быть плавной, задняя нога согнута, передняя нога прямая. Во время релиза не нужно толкаться задней ногой, она только держит упор. Приземление должно выполняться на ноги мягко и пружинисто, амортизируя приземление.
youtube	Ссылка на видеоурок	Символьный	255	‘А-я’, ‘ ’, ‘A-z’	нет		DLrGta3P_dY
save_video	Ссылка на веб-сервис для сохранения видеоурока	Символьный	255	‘А-я’, ‘ ’, ‘A-z’	нет		https://vk.com/away.php?to=https%3A%2F%2Fwww.clipconverter.cc%2Fdownload%2FXSf9yCog%2F480040790%2F&el=snippet

Таблица Б.2 – Атрибуты сущности «point»

Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
Id	Номер маркера	Целый	4	0001-9999	нет	PK	1
name	Название маркера	Символьный	255	‘А-я’, ‘ ’, ‘A-z’	нет		Wake Park Barbados
latitude	Широта	Символьный	255	‘А-я’, ‘ ’, ‘A-z’	нет		53.314982
longitude	Долгота	Символьный	255	‘А-я’, ‘ ’, ‘A-z’	нет		44.769030
addres	Адрес	Символьный	255	‘А-я’, ‘ ’, ‘A-z’	нет		Адрес: Пенза, Ул. Оздоровительная 2, 3 пляж
phone_number	Номер телефона	Символьный	255	‘А-я’, ‘ ’, ‘A-z’	нет		89648650656
phone_use_url	Ссылка на вызов	Символьный	255	‘А-я’, ‘ ’, ‘A-z’	нет		tel: 89648650656
one_set	Цена за сет	Символьный	255	‘А-я’, ‘ ’, ‘A-z’	нет		Разовое катание - 1 сет: 300 рублей
all_day	Цена за день катания	Символьный	255	‘А-я’, ‘ ’, ‘A-z’	нет		Целый день: 1000 рублей
image	Ссылка на изображение	Символьный	255	‘А-я’, ‘ ’, ‘A-z’	нет		https://vk.com/away.php?utf=1&to=http%3A%2F%2Fdrive.google.com%2Fuc%3Fexport%3Dview%26id%3D1ANuKEIdM3T_R2pZVF8ZE45MMdE23vKSt

Таблица Б.3 – Атрибуты сущности «question»

Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
Id	Номер вопроса	Целый	4	0001-9999	нет	PK	1
Id_lesson	Номер урока	Целый	4	0001-9999	нет	FK	2
question	Вопрос	Символьный	255	‘А-я’, ‘ ’, ‘A-z’	нет		На какой лебедке необходимо учить данный элемент?

Продолжение таблицы Б.3

version_first	Первый вариант ответа	Символьный	255	‘А-я’, ‘ ’, ‘A-z’	нет		На круговой
version_second	Второй вариант ответа	Символьный	255	‘А-я’, ‘ ’, ‘A-z’	нет		На реверсивной
version_third	Третий вариант ответа	Символьный	255	‘А-я’, ‘ ’, ‘A-z’	нет		Без разницы
true_version	Правильный вариант ответа	Символьный	255	‘А-я’, ‘ ’, ‘A-z’	нет		На реверсивной

Таблица Б.4 – Атрибуты сущности «competition»

Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
Id	Номер соревнования	Целый	4	0001-9999	нет	PK	1
name	Название соревнования	Символьный	255	‘А-я’, ‘ ’, ‘A-z’	нет		Первенство Белгородской области '20
date_comp	Дата проведения	Дата	8	Now – 01.01.2030	нет		20.06.2020
location	Месторасположения	Символьный	255	‘А-я’, ‘ ’, ‘A-z’	нет		Белгородская область
wakepark	Название вейк станции	Символьный	255	‘А-я’, ‘ ’, ‘A-z’	нет		Парк: Unity Wake Park
prizefond	Призовой фонд	Символьный	255	‘А-я’, ‘ ’, ‘A-z’	нет		Призовой фонд: 35000₽
startpay	Стартовый взнос	Символьный	255	‘А-я’, ‘ ’, ‘A-z’	нет		Стартовый взнос: 500₽
contacts	Контакты	Символьный	255	‘А-я’, ‘ ’, ‘A-z’	нет		Контактный адрес: belwake@mail.ru
registr	Ссылка на регистрацию	Символьный	255	‘А-я’, ‘ ’, ‘A-z’	нет		https://vk.com/away.php?utf=1&to=https%3A%2F%2Fwakebase.ru%2Fcomp%2F145%23registration
image	Ссылка на изображение	Символьный	255	‘А-я’, ‘ ’, ‘A-z’	нет		https://vk.com/away.php?utf=1&to=https%3A%2F%2Fdrive.google.com%2Fuc%3Fexport%3Dview%26id%3D12EMK7RCgs5Ki9vy9SAxcsvDxiPx-wDIQ

Приложение В.

ИСХОДНЫЙ КОД КЛАССОВ СЕРВЕРНОЙ ЧАСТИ

(рекомедуемое)

Листинг В.1 – Файл models.py

```
from peewee import *
import uuid

psql_db = PostgresqlDatabase(
'wakeboard_teacher',
user='wt_user',
password='123456',
host='127.0.0.1',
port=5432)

def init_tables():
psql_db.create_tables([Test, Lesson, Competition], safe=True)


def generate_points(num_points):
for i in range(num_points):
name = str(uuid.uuid4())[0:8]
latitude = str(uuid.uuid4())[0:8]
longitude = str(uuid.uuid4())[0:8]
addres = str(uuid.uuid4())[0:8]
phone_number = str(uuid.uuid4())[0:8]
phone_use_url = str(uuid.uuid4())[0:8]
one_set = str(uuid.uuid4())[0:8]
all_day = str(uuid.uuid4())[0:8]
image = str(uuid.uuid4())[0:8]
Point(name=name, latitude=latitude, longitude=longitude, addres=addres,
phone_number=phone_number, phone_use_url=phone_use_url, one_set=one_set,
all_day=all_day, image=image).save()

def generate_competitions(num_competitions):
for i in range(num_competitions):
name = str(uuid.uuid4())[0:8]
date_comp = datetime.date(uuid.uuid4())
location = str(uuid.uuid4())[0:8]
wakepark = str(uuid.uuid4())[0:8]
prizefond = str(uuid.uuid4())[0:8]
startpay = str(uuid.uuid4())[0:8]
contacts = str(uuid.uuid4())[0:8]
registr = str(uuid.uuid4())[0:8]
image = str(uuid.uuid4())[0:8]
Competition(name=name, date_comp=date_comp, location=location,
wakepark=wakepark, prizefond=prizefond, startpay=startpay, contacts=contacts,
registr=registr, image=image).save()

def generate_lessons(num_lessons):
for i in range(num_lessons):
name = str(uuid.uuid4())[0:8]
image = str(uuid.uuid4())[0:8]
level = str(uuid.uuid4())[0:8]
```

```
about = str(uuid.uuid4())[0:8]
youtube = str(uuid.uuid4())[0:8]
save_video = str(uuid.uuid4())[0:8]
Lesson(name=name, image=image, level=level, about=about, youtube=youtube,
save_video=save_video).save()

def generate_tests(num_tests):
for i in range(num_tests):
id_lesson = int(uuid.int())[0:8]
question = str(uuid.uuid4())[0:8]
version_first = str(uuid.uuid4())[0:8]
version_second = str(uuid.uuid4())[0:8]
version_third = str(uuid.uuid4())[0:8]
true_version = str(uuid.uuid4())[0:8]
Test(id_lesson=id_lesson, question=question, version_first=version_first,
version_second=version_second, version_third=version_third,
true_version=true_version).save()

class BaseModel(Model):
class Meta:
database = psql_db

class Point(BaseModel):
name = CharField()
latitude = CharField()
longitude = CharField()
addres = CharField()
phone_number = CharField()
phone_use_url = CharField()
one_set = CharField()
all_day = CharField()
image = CharField()

class Lesson(BaseModel):
name = CharField()
image = CharField()
level = CharField()
about = TextField()
youtube = CharField()
save_video = CharField()

class Competition(BaseModel):
name = CharField()
date_comp = DateField()
location = CharField()
wakepark = CharField()
prizefond = CharField()
startpay = CharField()
contacts = CharField()
registr = CharField()
image = CharField()
```

```
class Question(BaseModel):
id_lesson = ForeignKeyField(Lesson, backref='tweets')
question = CharField()
version_first = CharField()
version_second = CharField()
version_third = CharField()
true_version = CharField()
```

Листинг В.2 – Файл app.py

```
import falcon
from resources.point import PointDocument, PointsCollection
from resources.competition import CompetitionDocument, CompetitionsCollection
from resources.lesson import LessonDocument, LessonsCollection
from resources.question import QuestionDocument, QuestionsCollection

api = falcon.API()

api.add_route('/points/', PointsCollection())
api.add_route('/points/{point_id}', PointDocument())
api.add_route('/competitions/', CompetitionsCollection())
api.add_route('/competitions/{competition_id}', CompetitionDocument())
api.add_route('/lessons/', LessonsCollection())
api.add_route('/lessons/{lesson_id}', LessonDocument())
api.add_route('/questions', QuestionsCollection())
api.add_route('/questions/{id_lesson}', QuestionDocument())
```

Листинг В.3 – Файл competition.py

```
import falcon
from models import Competition
from playhouse.shortcuts import model_to_dict
import json

class CompetitionsCollection():
def on_get(self, req, resp):
competitions = Competition.select().order_by(Competition.id)
resp.body = json.dumps([model_to_dict(competition) for competition in competitions])

class CompetitionDocument():
def on_get(self, req, resp, competition_id):
competition = Competition.get(Competition.id == competition_id)
resp.body = json.dumps(model_to_dict(competition))
```

Листинг В.4 – Файл lesson.py

```
import falcon
from models import Lesson
from playhouse.shortcuts import model_to_dict
import json
```

```
class LessonsCollection():
def on_get(self, req, resp):
lessons = Lesson.select().order_by(Lesson.id)
resp.body = json.dumps([model_to_dict(lesson) for lesson in lessons])

class LessonDocument():
def on_get(self, req, resp, lesson_id):

lesson = Lesson.get(Lesson.id == lesson_id)
resp.body = json.dumps(model_to_dict(lesson))
```

Листинг В.5 – Файл point.py

```
import falcon
from models import Point
from playhouse.shortcuts import model_to_dict
import json

class PointsCollection():
def on_get(self, req, resp):
points = Point.select().order_by(Point.id)
resp.body = json.dumps([model_to_dict(point) for point in points])

class PointDocument():
def on_get(self, req, resp, point_id):

point = Point.get(Point.id == point_id)
resp.body = json.dumps(model_to_dict(point))
```

Листинг В.6 – Файл question.py

```
import json
import falcon
from playhouse.shortcuts import model_to_dict
from webargs import fields
from webargs.falconparser import use_args

from models import Question

class QuestionsCollection():

req_args = {
'lesson_id': fields.Integer()
}

@use_args(req_args, location="query")
def on_get(self, req, resp, validated_args):
questions = Question.select().order_by(Question.id)

if validated_args.get('lesson_id'):
```

```
questions = questions.where((Question.lesson_id == validated_args['lesson_id']))

resp.body = json.dumps([model_to_dict(question) for question in questions])

class QuestionDocument():
def on_get(self, req, resp, id_lesson):

question = Question.select(Question).where(Question.id_lesson == id_lesson)
resp.body = json.dumps(model_to_dict(question))
```

Приложение Г

ИСХОДНЫЙ КОД МОБИЛЬНОГО ПРИЛОЖЕНИЯ

(рекомендуемое)

Листинг Г.1 – Файл index.js

```
import {AppRegistry} from 'react-native';
import App from './App.js';
import {name as appName} from './app.json';
import {YellowBox} from 'react-native';

YellowBox.ignoreWarnings([
    'Remote debugger',
    'VirtualizedLists should never be nested',
    'componentWillReceiveProps has been renamed',
    'componentWillMount has been renamed',
    'Each child in a list',
    'Failed child context'
])
AppRegistry.registerComponent(appName, () => App);
```

Листинг Г.2 – Файл App.js

```
import React, {Component} from 'react';
import AppNav from './src/AppNav';

export default class App extends Component{
  render() {
    return (
      // <Provider {...stores}>
        // <StyleProvider style={getTheme(variables)}>
          <AppNav/>
        // </StyleProvider>
      // </Provider>
    );
  }
}
```

Листинг Г.3 – Файл AppNav.js

```
import React, {Component} from 'react';
import { createAppContainer, createSwitchNavigator } from 'react-navigation';

import PrivateNavigator from './screens/privateScope'


const AppNavigator = createAppContainer(
  createSwitchNavigator(
    {
      PrivateScope: PrivateNavigator
    },
    {
      initialRouteName: 'PrivateScope',
      headerMode: 'none'
```

```
    }
  )
);


export default class AppNav extends Component {
  render() {
    return (
      <AppNavigator />
    );
  }
}
```

Листинг Г.4 – Файл Api.js

```
import axios from 'axios';
const API_HOST = "http://wbt.localhost.com/"; //"http://192.168.1.35:8000/"


export default class Api {

  static async getHeaders(method) {

    const contentType = { 'Content-Type': 'application/json' };

    return {
      ...contentType
    }
  }

  static get(url, config) {
    return this.doRequest(url, config, 'get');
  }

  static put(route, config) {
    return this.doRequest(route, config, 'put')
  }

  static post(route, config) {
    return this.doRequest(route, config, 'post')
  }

  static delete(route, config) {
    return this.doRequest(route, config, 'delete')
  }

  static async doRequest(url, conf, method) {
    const config = Object.assign({
      headers: await this.getHeaders(method),
      method: method,
      baseURL: API_HOST,
      url: url
```

```
    }, conf);

    return axios(config);
  }

}
```

Листинг Г.5 – Файл src/screens/privateScope/index.js

```
import React from 'react';
import { createBottomTabNavigator } from 'react-navigation-tabs';
import Icon from 'react-native-vector-icons/FontAwesome';
//import IconEntypo from 'react-native-vector-icons/Entypo';

import LessonsNavigator from './lessons'
import ProfileNavigator from './profile'
import AboutWBNavigator from './aboutWB'
import CompetitionNavigator from './competition';
import LocationNavigator from './locationparks';
//import EquipmentNavigator from './equipment';


export default (PrivateNavigator = createBottomTabNavigator(
  {
    Profile: {
      screen: ProfileNavigator,
      navigationOptions: {
        tabBarLabel: 'Тренер',
        tabBarIcon: ({ tintColor }) => (
          <Icon
            name='male'
            size={24}
            style={{ color: tintColor }}
          />
        )
      }
    },
    Lessons: {
      screen: LessonsNavigator,
      navigationOptions: {
        tabBarLabel: 'Уроки',
        tabBarIcon: ({ tintColor }) => (
          <Icon
            name='book'
            size={24}
            style={{ color: tintColor }}
          />
        )
      }
    },
    AboutWB: {
      screen: AboutWBNavigator,
```

```
      navigationOptions: {
        tabBarLabel: 'Вейкбординг',
        tabBarIcon: ({ tintColor }) => (
          <Icon
            name='info'
            size={24}
            style={{ color: tintColor }}
          />
        )
      }
    },
    Competition: {
      screen: CompetitionNavigator,
      navigationOptions: {
        tabBarLabel: 'Соревнования',
        tabBarIcon: ({ tintColor }) => (
          <Icon
            name='trophy'
            size={24}
            style={{ color: tintColor }}
          />
        )
      }
    },
    Location: {
      screen: LocationNavigator,
      navigationOptions: {
        tabBarLabel: 'Вейкпарки',
        tabBarIcon: ({ tintColor }) => (
          <Icon
            name='map-marker'
            size={24}
            style={{ color: tintColor }}
          />
        )
      }
    },
  },
));
```

Листинг Г.6 – Файл src/screens/privateScope/profile/index.js

```
import { createStackNavigator } from 'react-navigation-stack';

import ProfileScreen from './ProfileScreen';

export default (LessonsNavigator = createStackNavigator(
  {
    Profile: { screen: ProfileScreen },
  },
  {
    initialRouteName: 'Profile',
```

```
  }
));
```

Листинг Г.7 – Файл ProfileScreen.js

```
import React, { Component, useCallback  } from 'react';
import {
    Text,
    View,
    Image,
    ScrollView,
    SafeAreaView,
    StyleSheet,
    Dimensions,
    Linking,
    Button,
    Alert,
} from 'react-native';
import { Colors, } from 'react-native/Libraries/NewAppScreen';
import { Icon } from 'native-base';

export default class ProfileScreen extends Component {

    render() {
        return (
            <SafeAreaView>
                <ScrollView
                    contentInsetAdjustmentBehavior="automatic"
                    style={{ backgroundColor: Colors, }}>
                    <View>
                        <View
                            style={styles.viewTitle}>
                            <Text style={styles.textTitle}>Обучение проводит
профессиональный тренер</Text>
                        </View>
                        <View >
                            <Image
                                style={styles.imageStyle}
                                source={require('./img/teacher2.jpg')} />
                            <Text style={styles.nameText}>
                                Лысенко Владимир Викторович
                            </Text>
                        </View>
                        <View style={{paddingBottom: 10,}}>
                            <Text style={styles.textStyle}>{'\t'}
                                Владимир родился в Днепропетровске (Украина)
в 1984 году.
                                Он начал катание на вейкборде в 2008 году. За
время катания
                                Владимир обрел огромный багаж знаний, которым
с радостью делиться
```

```
                                с любым желающим. Он проводит обучение в
PhuketWakePark на Пхукете.
                                Каждое лето Владимир путешествует по России и
проводит мастер классы,
                                подходящие для людей с разным уровнем
катания.</Text>
                                <Text style={styles.textStyle}>{'\t'}Кроме
того, Владимир
                                проводит удаленные онлайн тренировки и
удаленные мастер классы. Он так же
                                является создателем обучающего канала на
YouTube.
                                Владимир является
                                действующим чемпионом Украины по версиям WWA
                                и IFFW.
                            </Text>
                            <Text style={styles.textStyle}>{'\t'}
                                WWA (World Wake Association) - Мировая
ассоциация вейкбордистов
                                это органицзация, созданная
катающимися(райдерами) в 1989 году, которая
                                занимается продвижением и развитием вейк-
спорта по всему миру.
                                Ассоциация является основным мировым
источником стандартов соревнований,
                                а так же является бесспорным санкционным
органом управления.
                            </Text>
                            <Text style={styles.textStyle}>{'\t'}
                                IFFW (International Waterski and Wakeboard
Federation) - Международная
                                федерация катания на водных лыжах и
вейкборде. Она является мировым руководящим
                                органом для всех видов водного спорта на
буксире.
                            </Text>
                        </View>

                        <View style={{flexDirection: "row",
alignItems:"center", paddingBottom: 5,}}>
                            <Icon
                            name='logo-vk'
                            size={24}
                            style={{paddingLeft:10,
                            color: "#597da3"}} />
                            <OpenURLButton url={vkURL}
style={styles.iconTextStyle}>Задай вопрос тренеру</OpenURLButton>
                        </View>


                        <View style={{flexDirection: "row",
alignItems:"center",}}>
                        <Icon
```

```
                                name='logo-youtube'
                                size={24}
                                style={{paddingLeft:10, color:"red"}} />
                                <OpenURLButton url={youtubeURL}
style={styles.iconTextStyle}>YouTube канал "Wakeboard
Teacher"</OpenURLButton>
                            </View>
                        </View>
                    </ScrollView>
                </SafeAreaView>
            );
        }
}

const styles = StyleSheet.create({
    viewTitle: {
        paddingTop: 5,
        alignItems: "center",
        paddingBottom: 30,
    },
    textTitle: {
        fontSize: 23,
        textAlign: "center"
    },
    imageStyle: {
        width: windowWidth,
        height: 360,
        marginLeft: 7,
        marginRight: 7,
    },
    nameText: {
        paddingTop: 10,
        textAlign: "center",
        fontSize: 16,
        paddingBottom: 15,
    },
    textStyle: {
        fontSize: 16,
        paddingLeft: 10,
        paddingRight: 10,
    },
    iconTextStyle: {
        fontSize: 10,
        paddingLeft: 10,
        paddingRight: 10,
    }


});
const windowWidth = Dimensions.get('window').width;

const vkURL = "https://vk.com/bittybest";
```

```
const youtubeURL =
"https://www.youtube.com/channel/UCJesnl5X5SheAqpjbZoTrEA";
const whatsappURL = "whatsapp://send?text=hello&phone=89996119107"

const OpenURLButton = ({ url, children }) => {
    const handlePress = useCallback(async () => {
        const supported = await Linking.canOpenURL(url);
        if (supported) {
            await Linking.openURL(url);
        } else {
            Alert.alert(`Don't know how to open this URL: ${url}`);
        }
    }, [url]);

    return <Button title={children} onPress={handlePress} />;
};
```

Листинг Г.8 – файл src/screens/privateScope/lessons/index.js

```
import { createStackNavigator } from 'react-navigation-stack';

import ListScreen from './ListScreen';
import LessonScreen from './LessonScreen';
import FirstDay from './FirstDay';
import HS_Raley from './HS_Raley';;

export default (LessonsNavigator = createStackNavigator(
  {
    LessonsList: { screen: ListScreen },
    Lesson: { screen: LessonScreen },
  },
  {
    initialRouteName: 'LessonsList',

  }
));
```

Листинг Г.9 – файл ListScreen.js

```
import React, { Component } from 'react';
import {
  Text,
  View,
  SafeAreaView,
  ScrollView,
  StyleSheet,
  Image,
  Dimensions,
  TouchableHighlight
} from 'react-native';
import { Colors, } from 'react-native/Libraries/NewAppScreen';
```

```
import { FlatList } from 'react-native-gesture-handler';
import {
  Body,
  ListItem,
  Right,
} from 'native-base';
import Api from '../../../lib/Api';

export default class ListScreen extends Component {

  state = {
    lessons: null
  }

  componentDidMount() {
    this.loadData();
  }

  async loadData() {
    try {
      const response = await Api.get(`/lessons`);
      this.setState({
        lessons: response.data
      });
    } catch (error) {
      console.error(error);
    }
  }

  onPressItem = (lessonId) => {
    this.props.navigation.navigate('Lesson', { lessonId });
  }

  renderItem = ({ item }) => (
    <ListItem last
      //   style={styles.listItem}
      style={{
        backgroundColor: "#F5F5F5",
        borderBottomWidth: 1,
        borderTopWidth: 1,
      }}
      onPress={() => this.onPressItem(item.id)}
    >
      <View style={styles.viewImage}>
        <Image
          style={styles.imageStyle}
          source={{ uri: item.image }} />
      </View>
      <View style={styles.viewText}>
        <Text style={styles.textStyle}>
          {item.name}
        </Text>
        <Text style={{ paddingRight: 15, }}>
```

```
            {item.level}
          </Text>
        </View>
      </ListItem>

    );

    render() {
      return (
        <SafeAreaView>
          <ScrollView
            contentInsetAdjustmentBehavior="automatic"
            style={{ backgroundColor: Colors.lighter, height: windowHeight }}>
            <FlatList
              data={this.state.lessons}
              renderItem={this.renderItem}
              style={{ marginBottom: 170 }}
              keyExtractor={item => item.id}
            />
          </ScrollView>
        </SafeAreaView>
      );
    }
}

const windowWidth = Dimensions.get('window').width;
const windowHeight = Dimensions.get('window').height;

const styles = StyleSheet.create({
  viewMain: {
    flexDirection: "row",
    height: 70,
    backgroundColor: "#F5F5F5",
    borderBottomWidth: 1,
    borderTopWidth: 1,
    borderTopColor: "#C0C0C0",
    borderBottomColor: "#C0C0C0",
  },
  viewImage: {
    justifyContent: "center",
    alignItems: "center",
    flex: 0.8,
  },
  imageStyle: {
    width: 80,
    height: 36,
  },
  viewText: {
    justifyContent: "center",
    alignItems: "flex-start",
    flex: 2.2,
  },
  textStyle: {
```

```
    fontSize: 17,
    paddingRight: 15,
  },
  button: {
    backgroundColor: 'white',
  },
})
```

Листинг Г.10 – файл LessonScreen.js

```
import React, { Component, useCallback } from 'react';
import {
    Text,
    View,
    SafeAreaView,
    ScrollView,
    StyleSheet,
    TouchableOpacity,
    WebView,
    Button,
    Alert,
    Linking,
} from 'react-native';
import { Colors, } from 'react-native/Libraries/NewAppScreen';
import YouTube from 'react-native-youtube';
import { Drawer } from 'native-base';
import SideBar from "./Test";

import Api from '../../../lib/Api';

const OpenURLButton = ({ url, children }) => {
    const handlePress = useCallback(async () => {
        const supported = await Linking.canOpenURL(url);
        if (supported) {
            await Linking.openURL(url);
        } else {
            Alert.alert(`Don't know how to open this URL: ${url}`);
        }
    }, [url]);

    return <Button title={children} onPress={handlePress} />;
};

export default class FirstDay extends Component {

    state = {
        lesson: null
    }

    componentDidMount() {
        const lessonId = this.props.navigation.getParam('lessonId');
        this.loadData(lessonId);
```

```
    }

    async loadData(lessonId) {
        try {
            const response = await Api.get(`/lessons/${lessonId}`);
            this.setState({
                lesson: response.data
            });
        } catch (error) {
            console.error(error);
        }
    }

    onTextPress() {

        this.openDrawer();
    }

    closeDrawer = () => {
        this.drawer._root.close()
    }

    openDrawer = () => {
        this.drawer._root.open()
    }

    render() {
        return (
            <Drawer
                ref={(ref) => { this.drawer = ref; }}
                side='bottom'
                openDrawerOffset={(viewport) => viewport.height - 600} //
Отступ сверху, 600 высота PointPreviewPanel
                panCloseMask={0.3} // Место с которого можно смахнуть панель
                content={
                    <SideBar navigator={this.navigator}
                        lesson={this.state.lesson && this.state.lesson} />}
                onClose={() => this.closeDrawer()}>
                <SafeAreaView>
                    <ScrollView
                        contentInsetAdjustmentBehavior="automatic"
                        style={{ backgroundColor: Colors.lighter, }}>
                        <View style={styles.titleView}>
                            <Text style={styles.titleText}>
                                {this.state.lesson && this.state.lesson.name}
                            </Text>
                        </View>
                        <View style={styles.videoView}>
                            <YouTube
                                videoId={this.state.lesson &&
this.state.lesson.youtube}
                                play
                                fullscreen
```

```
                                loop
                                onReady={e => this.setState({ isReady: true
})}
                                onChangeState={e => this.setState({ status:
e.state })}
                                onChangeQuality={e => this.setState({
quality: e.quality })}
                                onError={e => this.setState({ error: e.error
})}
                                style={{ alignSelf: 'stretch', height: 300 }}
                            />
                        </View>
                        <View>
                            <Text style={styles.textAboutEl}>
                                {this.state.lesson &&
this.state.lesson.about}
                            </Text>
                        </View>
                        <TouchableOpacity onPress={(e) =>
this.onTextPress(this.state.lesson && this.state.lesson)}>
                            <View style={{ paddingTop: 10, paddingBottom: 10
}}>
                                <Text style={styles.testStyle}>Пройти
тестирование</Text>
                            </View>
                        </TouchableOpacity>
                        <TouchableOpacity>
                            <View>
                                <OpenURLButton url={this.state.lesson &&
this.state.lesson.save}>Скачать видео</OpenURLButton>
                            </View>
                        </TouchableOpacity>
                    </ScrollView>
                </SafeAreaView>
            </Drawer>
        );
    }
}

const styles = StyleSheet.create({
    titleView: {
        paddingTop: 10,
        paddingBottom: 10,

    },
    titleText: {
        fontSize: 20,
        textAlign: "center",
    },
    videoView: {
        paddingLeft: 5,
        paddingRight: 5,
    },
```

```
    textAboutEl: {
        paddingTop: 10,
        paddingLeft: 5,
        paddingRight: 5,
        fontSize: 17,

    },
    testStyle: {
        textAlign: "center",
        color: "#1E90FF",
        fontSize: 18
    }

})
```

Листинг Г.11 – файл Test.js

```
import PropTypes from 'prop-types';
import React, { Component } from 'react';
import { NavigationActions } from 'react-navigation';
import { ScrollView, Text, View, Dimensions, SafeAreaView, Image, FlatList,
StyleSheet, Button, Alert } from 'react-native';
import { StackNavigator } from 'react-navigation';
import { Linking } from 'react-native';
import Api from '../../../lib/Api';

import { Container, Header, Content, ListItem, Radio, Right, Left } from
'native-base';
import { TouchableOpacity } from 'react-native-gesture-handler';

let i = 0

class SideMenu extends Component {

    constructor() {
        super()
        this.state = {
            data: [
                { "id": "0", "title": "Нужно ли амортизировать коленями при
приземлении?", "Var1": 'Да', "Var2": 'Вперед', "Var3": 'Посмотреть вперед' },
                { "id": "1", "title": "Куда нужно смотреть при приземлении?",
"Var1": 'Нет', "Var2": 'На доску', "Var3": 'Посмотреть через переднее плечо'
},
                { "id": "2", "title": "Как инициировать вращение после
релиза?", "Var1": 'По желанию', "Var2": 'На дальнюю опору', "Var3":
'Посмотреть через заднее плечо' }
            ],
            checkedOne: 0,
            checkedSecond: 0,
            checkedThird: 0,
            key: 0,
        }
```

```
    }

    navigateToScreen = (route) => () => {
        const navigateAction = NavigationActions.navigate({
            routeName: route
        });
        this.props.navigation.dispatch(navigateAction);
    }

    onPressPoint = ({ key }) => {
        this.setState({ checkedOne: key })
        if (key == this.state.data.Var3) {
            i++
        }
        alert(i)
    }


    render() {
        return (
            <Container>
                <View>
                    <Text style={styles.testText}>Тестирование</Text>
                    <Text>{this.state.data[0].title}</Text>
                    {
                        this.state.data.map((data, key) => {
                            return (
                                <View>
                                    {
                                        this.state.checkedOne == key ?
                                            <TouchableOpacity
style={styles.btn}>
                                                <Image style={styles.img}
source={require("./img/selectedBtn.png")} />
                                                <Text>{data.Var1}</Text>
                                            </TouchableOpacity>
                                            :
                                            <TouchableOpacity
style={styles.btn} onPress={() => { this.setState({ checkedOne: key }) }}>
                                                <Image style={styles.img}
source={require("./img/btn.png")} />
                                                <Text>{data.Var1}</Text>
                                            </TouchableOpacity>
                                    }
                                </View>
                            )
                        })
                    }
                    <Text>{this.state.data[1].title}</Text>
                    {
                        this.state.data.map((data, key) => {
                            return (
                                <View>
```

```
                                    {
                                        this.state.checkedSecond == key ?
                                            <TouchableOpacity
style={styles.btn}>
                                                <Image style={styles.img}
source={require("./img/selectedBtn.png")} />
                                                <Text>{data.Var2}</Text>
                                            </TouchableOpacity>
                                            :
                                            <TouchableOpacity
style={styles.btn} onPress={() => { this.setState({ checkedSecond: key }) }}>
                                                <Image style={styles.img}
source={require("./img/btn.png")} />
                                                <Text>{data.Var2}</Text>
                                            </TouchableOpacity>
                                    }
                                </View>
                            )
                        })
                    }
                    <Text>{this.state.data[2].title}</Text>
                    {
                        this.state.data.map((data, key) => {
                            return (
                                <View>
                                    {
                                        this.state.checkedThird == key ?
                                            <TouchableOpacity
style={styles.btn}>
                                                <Image style={styles.img}
source={require("./img/selectedBtn.png")} />
                                                <Text>{data.Var3}</Text>
                                            </TouchableOpacity>
                                            :
                                            <TouchableOpacity
style={styles.btn} onPress={() => { this.setState({ checkedThird: key }) }}>
                                                <Image style={styles.img}
source={require("./img/btn.png")} />
                                                <Text>{data.Var3}</Text>
                                            </TouchableOpacity>
                                    }
                                </View>
                            )
                        })
                    }
                    <Button title="Протестировать"
                        onPress={() => Alert.alert("Пора приступать к
практике!")} />
                </View>
            </Container>
        );
    }
}
```

```
SideMenu.propTypes = {
    navigation: PropTypes.object
};

export default SideMenu;

let { height, width } = Dimensions.get('window')
const styles = StyleSheet.create({
    testText: {
        fontSize: 22,
        textAlign: "center",
        paddingTop: 10,
        paddingBottom: 10
    },
    questionText: {
        fontSize: 17,
        paddingTop: 10,
        paddingBottom: 15,
        paddingLeft: 10
    },
    question: {
        fontSize: 17,
    },
    img: {
        height: 20,
        width: 20,
        marginRight: 5
    },
    btn: {
        flexDirection: "row",
        alignItems: "center",
        marginLeft: 10,
        marginBottom: 10
    }
})
```

Листинг Г.12 – файл src/screens/privateScope/aboutWB/index.js

```
import { createStackNavigator } from 'react-navigation-stack';

import AboutWBScreen from './AboutWBScreen';

export default (LessonsNavigator = createStackNavigator(
  {
    AboutWB: { screen: AboutWBScreen},
  },
  {
    initialRouteName: 'AboutWB',
  }
));
```

Листинг Г.13 – файл AboutWBScreen.js

```
import React, { Component } from 'react';
import {
  Text,
  Image,
  Dimensions,
  StyleSheet,
  SafeAreaView,
} from 'react-native';
import { Container, Header, Content, Accordion, View, Icon } from "native-
base";


export default class AboutWBScreen extends Component {

  _renderHeader(item) {
    switch (dataArray.indexOf(item)) {
      case 0:
        return (
          <View style={{
            height: 50,
            backgroundColor: "#DCDCDC",
            borderWidth: 1,
            borderColor: "black",
            borderRadius: 10,
            marginTop: 5,
            marginBottom: 5,
            alignItems: "center",
            justifyContent: "center",
          }}>
            <Text style={{ fontSize: 20, }}>{item.title}</Text>
          </View>
        );
        break;
      case 1:
        return (
          <View style={{
            height: 50,
            backgroundColor: "#DCDCDC",
            borderWidth: 1,
            borderColor: "black",
            borderRadius: 10,
            marginTop: 5,
            marginBottom: 5,
            alignItems: "center",
            justifyContent: "center",
          }}>
            <Text style={{ fontSize: 20, }}>{item.title}</Text>
          </View>
        );
        break;
```

```
      case 2:
        return (
          <View style={{
            height: 50,
            backgroundColor: "#DCDCDC",
            borderWidth: 1,
            borderColor: "black",
            borderRadius: 10,
            marginTop: 5,
            marginBottom: 5,
            alignItems: "center",
            justifyContent: "center",
          }}>
            <Text style={{ fontSize: 20, }}>{item.title}</Text>
          </View>
        );
        break;
      case 3:
        return (
          <View style={{
            height: 50,
            backgroundColor: "#DCDCDC",
            borderWidth: 1,
            borderColor: "black",
            borderRadius: 10,
            marginTop: 5,
            marginBottom: 5,
            alignItems: "center",
            justifyContent: "center",
          }}>
            <Text style={{ fontSize: 20, }}>{item.title}</Text>
          </View>
        );
        break;
    }
  }

  _renderContent(item) {
    switch (dataArray.indexOf(item)) {
      case 0:
        return (
          <SafeAreaView>
            <View>
              <View >
                <Text style={styles.TextStyle}>
                  {'\t'}Вейкбординг - это молодой вид спорта, основанный как
на водных лыжах, так и на серфинге и сноуборде.
                </Text>
                <Text style={styles.TextStyle}>{'\t'}На протяжении многих лет
утверждается, что вейкбординг возник в разных местах.
                  Наиболее широко признанное происхождение происходит в
Австралии в 1980-х годах. Джефф Дарби и его друзья занялись созданием
собственных досок для серфинга, специально предназначенных для буксировки за
```

```
лодками. В итоге они создали собственную торговую марку Skurfer, которая
продавала эти доски.
                </Text>
                <Text style={styles.TextStyle}>{'\t'}Примерно в то же время
во Флориде, серфер по имени Говард Джейкобс занимался серфингом на своих
небольших досках для серфинга за моторными лодками, добавляя на доску ремни
для ног, чтобы совершать трюки, которые невозможно сделать на обычной доске
для серфинга.
                </Text>
                <Text style={styles.TextStyle}>{'\t'}К середине восьмидесятых
было два известных бренда, продающих вейкборды. Дарби объединил усилия с
создателем доски для серфинга и изобретателем Брюсом Макки, чтобы создать
доску «Макски».
                </Text>
                <Text style={styles.TextStyle}>{'\t'}Вейкбординг был добавлен
как соревновательный вид спорта в X Games II.
            </Text>
              </View>
              <View style={{
                alignItems: "center",
                paddingTop: 10,
                paddingBottom: 10
              }}>
                <Image
                  source={require('./img/history.jpg')}
                  style={{
                    width: windowWidth,
                    height: 400,
                  }}
                />
              </View>
            </View>
          </SafeAreaView>
        );
        break;
      case 1:
        return (
          <SafeAreaView>
            <View >
              <Text style={styles.titleText}>Не бойтесь упасть</Text>
              <Text style={styles.TextStyle}>{'\t'}Каждому начинающему
вейкбордисту необ&shy;ходимо знать, что ему предстоит много падений.
Инструктор по вейкбордингу скорее всего знает, что пошло не так. Внимательно
выслушайте все советы, которые он вам даст. Если что-то не поня&shy;ли –
переспросите!
            </Text>

              <Text style={styles.titleText}>Уставшие руки</Text>
              <Text style={styles.TextStyle}>{'\t'}На первом уроке по
вейкбордингу для начи&shy;нающих я заметил, как тяжело новичкам
удер&shy;живать ручку фала продолжительное время. Это абсолютно нормально. Вы
станете сильнее и при&shy;выкните к этому.
            </Text>
```

```
              <Text style={styles.titleText}>Выбор стойки</Text>
              <Text style={styles.TextStyle}>{'\t'}Для того чтобы узнать
какой ногой вперед вы будете кататься, необходимо разбежаться и резко
остановиться, выставив ногу вперед, которая и является ответом на вопрос.
Люди катающиеся левой ногой вперед - регуляры, правой - гуфи.
            </Text>
              <Text style={styles.titleText}>Длина фала</Text>
              <Text style={styles.TextStyle}>{'\t'}Для начинающих лучше
использовать короткие фалы,
            ручки которых едва касаются воды.
            </Text>
              <Text style={styles.titleText}>Не тяните фал на себя</Text>
              <Text style={styles.TextStyle}>{'\t'}Во время старта и катания
не тяните ручку фа&shy;ла на себя. Это не подтягивания! Следует помнить, что
за ручку фала следует держаться полусогну&shy;тыми руками и ни в коем случае
не тянуть ее на себя и не тянуться за ней!
            </Text>
              <Text style={styles.titleText}>Держите ручку фала у
бедра</Text>
              <Text style={styles.TextStyle}>{'\t'}Во время катания
старайтесь держать ручку фала у бедра, это позволит вам контролировать тягу.
            </Text>
              <Text style={styles.titleText}>Смотрите прямо перед
собой</Text>
              <Text style={styles.TextStyle}>{'\t'}Во время катания не стоит
смотреть на крепления, смотрите вперед. Это придаст вам уверенности в
маневрах.
          </Text>
              <Text style={styles.titleText}>Главное</Text>
              <Text style={styles.TextStyle}>{'\t'}Запомните главное!
Вейкбординг в первую очередь должен приносить вам радость и море
положительных эмоций!</Text>
            </View>
          </SafeAreaView>
        );
        break;
      case 2:
        return (
          <SafeAreaView>
            <View>
              <Text style={styles.TextStyle}>{'\t'}1. Катание без разрешения
и надзора инструктора строго запрещено. </Text>
              <Text style={styles.TextStyle}>{'\t'}2. Во время занятий
занимающиеся должны неукоснительно выполнять все требования и рекомендации
инструктора.
         В случае нарушения правил поведения и техники безопасности на воде
администрация вейк парка не несет ответственности за возможные
неблагоприятные последствия для вас и других гостей.</Text>
              <Text style={styles.TextStyle}>{'\t'}3. Запрещено катание во
время грозы. </Text>
              <Text style={styles.TextStyle}>{'\t'}4. Запрещено купание в
зоне катания. </Text>
```

```
              <Text style={styles.TextStyle}>{'\t'}5. Запрещен выход на воду
в алкогольном и наркотическом опьянении. </Text>
              <Text style={styles.TextStyle}>{'\t'}6. Запрещен выход на воду
без спасательного жилета и шлема.</Text>
              <Text style={styles.TextStyle}>{'\t'}7. Не рекомендуется
переживать и бояться! </Text>
              <Text style={styles.TextStyle}>{'\t'}8. К катанию допускаются
лица, умеющие плавать. </Text>
              <Text style={styles.TextStyle}>{'\t'}9. Лица возрастом до 18
лет катаются под надзором родителей. </Text>
              <Text style={styles.TextStyle}>{'\t'}10. Рукоятку фала
необходимо держать только руками. Запрещается зажимать ее между ног. </Text>
              <Text style={styles.TextStyle}>{'\t'}11. Катание по фигурам на
досках с металлическим кантом запрещено.</Text>
            </View>
          </SafeAreaView>
        );
        break;
      case 3:
        return (
          <SafeAreaView>
            <View>
              <View>
                <Text style={styles.titleText}>Начиная</Text>
                <Text style={{ fontSize: 16, }}>
                  {'\t'}Если вы хотите научиться кататься на вейк&shy;борде,
то на первое время вейк парк может сдать вам в аренду все необходимое.
Выбирая место для катания, удостоверьтесь, что там есть подхо&shy;дящее
именно для вас снаряжение. Однако, если вы заинтересованы в развитии, то вам
необходи&shy;мо приобрести персональное оборудование.
                  Для подбора вейк снаряжения нужно будет сообщить: вес и
размер вашей ноги. Обязательно наличие спасательного жилета, уточните наличие
жилета вашего размера. Если на улице не жарко, жела&shy;тельно использовать
гидрокостюм. Особое вни&shy;мание придайте наличию шлема.
                  Кроме того, не лишним будет наличие полотенца, сухих сменных
вещей, а так же немного провизии, чтобы подкрепиться после катания.
              </Text>
                <Image source={require('./img/ParkObor.jpg')}
                  style={styles.ImageOborStyle} />
              </View>
              <View>
                <Text style={{ fontWeight: "bold", fontSize: 20, textAlign:
"center" }}>Доска</Text>
                <Text style={styles.TextStyle}>{'\t'}У каждого вейкборда есть
функциональные элементы, от которых зависят его характеристики. Практически
самым важным элементом доски является рокер, то есть изгиб, от хвоста до носа
доски. Они бывают разных видов и соответственно по-разному влияют на
характеристики езды.</Text>
                <Image source={require("./img/Rocker.jpg")}
                  style={styles.ImageOborStyle} />
                <Text style={styles.TextStyle}>{'\t'} При выборе доски,
знания о рокерах, несом&shy;ненно, помогут вам. Начнем с того что рокеры
```

```
подразделяют на две основные формы это бес&shy;прерывный и трехступенчатый
рокер.
              </Text>
                <Text style={styles.TextStyle}>{'\t'}Беспрерывным рокером
является прямой по&shy;перечный изгиб доски от хвоста до кончика носа. К
преимуществам можно отнести довольно мягкий и вполне предсказуемый выход с
волны и при приземлении в воду хорошее гашение удара. Хорошо подойдет для
трюков с флэта. Есть у него и недостатки. На доске с таким рокером, быстрей
приходит усталость, так как сопротивление воды у него больше. Так же на такой
доске меньше амп&shy;литуда прыжков.
            </Text>
                <Image source={require("./img/nopreriv.png")}
                  style={styles.ImageStyle} />
                <Text style={styles.TextStyle}>{'\t'}Из преимуществ
трехступенчатого рокера можно выделить, что доска с подобным рокером меньше
погружается в воду, а это дает большую скорость и лучшую маневренность. С
трехсту&shy;пенчатым рокером прыжки будут довольно амплитудные, как с волны,
так и с флэта. Мину&shy;сами будет приличная отдача при приземлении и низкая
стабильность доски. </Text>
                <Image source={require('./img/trehstup.png')}
                  style={styles.ImageStyle} />

                <Text style={styles.TextStyle}>{'\t'}Помимо этих двух типов
есть еще и гибридный рокер, то есть золотая середина между двумя предыдущими.
Доски такого типа имеют среднюю часть от беспрерывного рокера, а две крайние
части от трехступенчатого. Такое устройство называют золотой серединой, а
сами доски весь&shy;ма ценятся, хоть и увидеть их можно довольно редко.
              </Text>
                <Image source={require('./img/gibridrok.png')}
                  style={styles.ImageStyle}
                />
                <Text style={styles.TextStyle}>{'\t'}Для новичков следует
брать доски с беспрерывным рокером и тогда обучение пройдет быстрей. Д я тех,
кто уже умеет стоять на доске, но стремится развить свое мастерство, все
равно стоит брать беспрерывный рокер или же гибридный.
              </Text><Text style={styles.TextStyle}>{'\t'}В помощи выбора
длины доски прекрасно подойдет следующая таблица:
            </Text>
                <Image source={require("./img/tableSize.jpg")}
                  style={{
                    width: windowWidth,
                    marginLeft: "auto",
                    marginRight: "auto",
                  }} />
              </View>
              <View>
                <Text style={{ fontWeight: "bold", fontSize: 20, textAlign:
"center", paddingTop: 10, paddingBottom: 10, }}>Крепления</Text>
                <Text style={styles.TextStyle}>{'\t'}Наиболее известный вид
крепления Open-toe. Это крепление с открытым носом. Его преиму&shy;ществами
является гибкость размера, что позво&shy;лит кататься людям с примерно
одинаковым размером ног.</Text>
                <Image source={require('./img/openToe.jpg')}
```

```
                    style={styles.ImageOborStyle} />
                  <Text style={styles.TextStyle}>
                    {'\t'}Крепления с закрытым носом - Closed Toe. Имеет
конкретный размер и обладает более жестким внутренником, который делает
контроль над доской лучше. Находиться в таких креплениях теплей, а значит,
что можно кататься в них и в прохладное время.
                  </Text>
                    <Image source={require('./img/closeToe.jpg')}
                      style={styles.ImageOborStyle} />
                    <Text style={styles.TextStyle}>
                      {'\t'}Также крепления различаются по системам фиксации ног.
                    </Text>
                      <Text style={styles.TextStyle}>
                        {'\t'}Самая популярная система -
                      это фиксация ног с помощью шнурков, они могут быть как толстые,
так и тонкие.</Text>
                      <Image source={require('./img/shnurki.jpg')}
                        style={styles.ImageOborStyle} />
                      <Text style={styles.TextStyle}>
                        {'\t'}Ремни Velcro - быстрый и легкий способ крепления, но
они менее надежны по сравнению со шнурками.
                      </Text>
                        <Image source={require('./img/velco.jpg')}
                          style={styles.ImageOborStyle} />
                        <Text style={styles.TextStyle}>
                          {'\t'}Система Switch Hinge - откидная пятка, через которую
можно безболезненно вставлять и вынимать ногу из крепления.
                        </Text>
                          <Image source={require('./img/switchHinge.jpg')}
                            style={styles.ImageOborStyle} />
                          <Text style={styles.TextStyle}>
                            {'\t'}Существует также система крепления как на сноуборде.
                          </Text>
                            <Image source={require('./img/snowBoard.jpg')}
                              style={styles.ImageOborStyle} />
                            <Text style={styles.TextStyle}>
                              {'\t'}Одним из наиболее значимых показателей уровня
крепления, является его твердость, чем тверже ботинок тем лучше фиксация
ноги. Главное перед началом катания, стоит хорошо зафиксировать ноги, это
позволит вам избежать ненужных вывихов, а так же, таким путем вы дольше
сохраните свои крепления в рабочем состоянии.
                          </Text>
                            </View>
                            <View>
                              <Text style={{ fontWeight: "bold", fontSize: 20, textAlign:
"center", paddingTop: 10, paddingBottom: 10, }}>Шлем</Text>
                              <Text style={styles.TextStyle}>{'\t'}Шлемы являются ключевым
элементом обес&shy;печения безопасности. Все, от начинающих
вейк&shy;бордистов до опытных профессионалов, носят шлем. Это связано с тем,
что на вейкбординг яв&shy;ляется травмоопасным спортом, в котором
при&shy;сутствуют элементы фристайла и джиббинга (прыжки с трамплинов и
катание по фигурам).</Text>
```

```
                <Text style={styles.TextStyle}>{'\t'}Шлемы для вейкбординга
разработаны спе&shy;циально для водных видов спорта, поэтому лучше купить
настоящий шлем для вейкбординга, чем пытаться использовать свой велосипедный
шлем на воде. Они легкие, водостойкие и вентилиру&shy;емые, что позволяет
воде легко вытекать. Mystic делают отличные шлемы для вейкбординга, как Bern
и Liquid Force.
            </Text>
                <Image source={require('./img/slem.jpg')}
style={styles.ImageOborStyle} />
              </View>
              <View>
                <Text style={{ fontWeight: "bold", fontSize: 20, textAlign:
"center", paddingTop: 10, paddingBottom: 10, }}>Спасательный жилет</Text>
                <Text style={styles.TextStyle}>{'\t'}Как и шлем, спас жилет
является главным атрибутом катания на вейкборде. Он удерживает вас в воде,
пока вы ждете, когда лодка или лебедка поднимет вас из воды. Он также полезен
в чрезвычайных ситуациях. Если вы потеряете сознание, жилет удержит вас на
поверхности воды.
              </Text>
                <Text style={styles.TextStyle}>{'\t'}Спас жилеты для
вейкбординга отличаются от обычных спас жилетов размерами, что позволяет
беспрепятственно выполнять любые элементы акробатики. O'Brien , Hyperlite и
Liquid Force - это вспомогательные средства для плавучести и жилеты для
вейкбордистов.
            </Text>
                <Image source={require('./img/Gillet.jpg')}
style={styles.ImageOborStyle} />
              </View>
              <View>
                <Text style={{ fontWeight: "bold", fontSize: 20, textAlign:
"center", paddingTop: 10, paddingBottom: 10, }}>Гидрокостюм</Text>
                <Text style={styles.TextStyle}>{'\t'}
                Изначально такой костюм бы создан для дай&shy;веров, которые
продолжительное время плавают под водой. Для вейкбордистов существуют
совершенно иные технологии.
                </Text>
                <Text style={styles.TextStyle}>{'\t'}Покупая костюм стоит
обратить свое внимание на его эластичность, он должен хорошо тянуться и
быстро приходить в исходное положение при этом, не потеряв толщины и без
появления рас&shy;тяжек.
                Костюмы делаются из Неопрена и его ком&shy;бинаций. То есть в
участках где происходит актив&shy;ное движение, например, в локтях, ставится
встав&shy;ка из тонкого более толстого неопрена, чем в остальном костюме.
Поэтому толщину костюма могут обозначить 3,4 или 2,3 и так да ее. В
неко&shy;торых костюмах используется три толщины, в этом случае написано
будет, например 3/4/5.
                </Text>
                <Text style={styles.TextStyle}>{'\t'}Стоит отметить, что
гидрокостюм в местах спины, паха и груди до жен быть утолщен, для меньшей
потери тепла. Также обратите внимание на колени, там должны стоять резиновые
накладки и чем больше они, тем лучше. Сухой гидрокостюм идет толщенной от
пяти миллиметров и выше, он полностью прорезинен и герметичен. Имеет длинные
рукава.
```

```
                В основном все вейкбордисты катаются в мокрых гидрокостюмах,
толщиной от 1мм. до 6мм. Он может иметь короткие рукава. Мокрый гидрокостюм
именуется мокрым, так как он греет за счет воды, которая становится небольшой
прослойкой между телом и неопреном, поддерживая постоянную температуру.
            </Text>
                <Image source={require('./img/gidrocost.jpg')}
style={styles.ImageOborStyle} />
              </View>
            </View>
          </SafeAreaView>
        );
        break;
    }
  };



  render() {
    return (
      <Container>
        <Header><Text style={{ fontSize: 23 }}>Полезная
информации</Text></Header>
        <Content padder style={{width: windowWidth}}>
          <Accordion dataArray={dataArray}
            //animation={true}
            expanded={true}
            renderHeader={this._renderHeader}
            renderContent={this._renderContent}
            icon="ios-arrow-forward"
            expandedIcon="ios-arrow-down"
          />
        </Content>
      </Container>
    );
  }
}


const windowWidth = Dimensions.get('window').width
const dataArray = [
  { title: "История" },
  { title: "Советы для начинающих" },
  { title: "Техника безопасности" },
  { title: "Выбор оборудования" },
];

const styles = StyleSheet.create({
  ImageStyle: {
    width: windowWidth*0.8,
    marginLeft: "auto",
    marginRight: "auto",
  },
  TextStyle: {
```

```
    fontSize: 16,
  },
  ImageOborStyle: {
    width: windowWidth*0.8,
    height: 300,
    marginLeft: "auto",
    marginRight: "auto",
    marginTop: 10,
    marginBottom: 10,
  },
  titleText: {
    fontWeight: "bold",
    fontSize: 20,
    textAlign: "center",
    paddingTop: 5,
    paddingBottom: 5,
  }
})
```

Листинг Г.14 – файл src/screens/privateScope/competition/index.js

```
import { createStackNavigator } from 'react-navigation-stack';

import CompetitionListScreen from './CompetitionListScreen';
import CompetitionScreen from './CompetitionScreen';

export default (LessonsNavigator = createStackNavigator(
  {
    CompetitionList: { screen: CompetitionListScreen},
    Competition:{ screen: CompetitionScreen},
  },
  {
    initialRouteName: 'CompetitionList',
  }
));
```

Листинг Г.15 – файл CompetitionListScreen.js

```
import React, { Component } from 'react';
import {
    Text,
    View,
    SafeAreaView,
    ScrollView,
    StyleSheet,
    TouchableHighlight,
    Dimensions,
    FlatList
} from 'react-native';

import {
```

```
    Body,
    ListItem,
    Right,
} from 'native-base';

import { Colors, } from 'react-native/Libraries/NewAppScreen';
import Icon from 'react-native-vector-icons/FontAwesome';

import Api from '../../../lib/Api';


let day =[]
let point = '.'
export default class CompetitionListScreen extends Component {

    state = {
        competitions: null
    }

    componentDidMount() {
        this.loadData();
    }

    async loadData() {
        try {
            const response = await Api.get(`/competitions`);
            this.setState({
                competitions: response.data
            });
        } catch (error) {
            console.error(error);
        }
    }

    onPressItem = (competitionId) => {
        this.props.navigation.navigate('Competition', { competitionId });
    }

    renderItem = ({ item }) => (

        <ListItem last
            //   style={styles.listItem}
            style={{
                backgroundColor: "#F5F5F5",
                borderBottomWidth: 1,
                borderTopWidth: 1,
            }}
            onPress={() => this.onPressItem(item.id)}
        >
            <View style={{ flexDirection: "row", alignItems: "center", flex:
1, marginLeft: 15, }}>
                <View>
```

```
                    <Text style={{ fontSize: 45 }}>{item.date_comp.split('
')[0]}</Text>
                </View>
                <View>
                    <Text style={{ fontSize: 20 }}>{item.date_comp.split('
')[1]}</Text>
                    <Text style={{ fontSize: 20 }}>{item.date_comp.split('
')[2]}</Text>
                </View>
            </View>
            <View style={{ alignItems: "flex-start", flex: 2, }}>
                <Text style={styles.textStyle}>{item.name}</Text>
                <Text style={styles.textStyle}>{item.location}</Text>
            </View>
        </ListItem>

    );

    render() {
        return (
            <SafeAreaView>
                <ScrollView
                    contentInsetAdjustmentBehavior="automatic"
                    style={{ backgroundColor: Colors.lighter, height:
windowHeight }}>

                    <FlatList
                        data={this.state.competitions}
                        renderItem={this.renderItem}
                        style={{ marginBottom: 170 }}
                        keyExtractor={item => item.id}
                    />
                </ScrollView>
            </SafeAreaView>
        );
    }

}

const windowWidth = Dimensions.get('window').width
const windowHeight = Dimensions.get('window').height;
const styles = StyleSheet.create({
    viewMain: {
        flexDirection: "row",
        backgroundColor: "#F5F5F5",
        borderBottomWidth: 1,
        borderTopWidth: 1,
        borderTopColor: "#C0C0C0",
        borderBottomColor: "#C0C0C0",

    },
    viewText: {
        justifyContent: "center",
```

```
        alignItems: "flex-start",
        flex: 2.2,
    },
    textStyle: {
        paddingLeft: 10,
        fontSize: 18,
        paddingTop: 2.5,
        paddingBottom: 2.5,

    },
})
```

Листинг Г.16 – файл CompetitionScreen.js

```
import React, { Component, useCallback } from 'react';
import {
    Text,
    SafeAreaView,
    ScrollView,
    StyleSheet,
    Image,
    Dimensions,
    FlatList,
    Button,
    Linking,
} from 'react-native';
import { Colors, } from 'react-native/Libraries/NewAppScreen';
import { Accordion, Content, Container, View, Icon } from 'native-base';

import Api from '../../../lib/Api';

export default class CompetitionScreen extends Component {

    state = {
        competition: null
    }

    componentDidMount() {
        const competitionId =
this.props.navigation.getParam('competitionId');
        this.loadData(competitionId);
    }

    async loadData(competitionId) {
        try {
            const response = await Api.get(`/competitions/${competitionId}`);
            this.setState({
                competition: response.data
            });
        } catch (error) {
            console.error(error);
        }
```

```
    }

    render() {
        return (
            <Container>
                <Content>
                    <SafeAreaView>
                        <ScrollView
                            contentInsetAdjustmentBehavior="automatic"
                            style={{
                                backgroundColor: Colors.lighter,
                                height: windowHeight
                            }}>
                            <Text style={styles.nameCompetition}>
                                {this.state.competition &&
this.state.competition.name}</Text>
                                {this.state.competition &&  <Image
source={{uri: this.state.competition.image}}
                                style={styles.imageStyle} /> }
                            <Text style={styles.textStyle}>Дата:
{this.state.competition &&
                            this.state.competition.date_comp} </Text>
                                <Text style={styles.textStyle}>Место:
{this.state.competition && this.state.competition.location}</Text>
                                <Text style={styles.textStyle}>
                                    {this.state.competition &&
this.state.competition.wakepark}</Text>
                                <Text style={styles.textStyle}>{
                                this.state.competition &&
this.state.competition.prizefond}</Text>
                                <Text style={styles.textStyle}>
                                    {this.state.competition &&
this.state.competition.startpay}</Text>
                                <Text style={styles.textStyle}>
                                    {this.state.competition &&
this.state.competition.contacts}</Text>
                            <OpenURLButton url={this.state.competition &&
this.state.competition.regist}
                            style={{ paddingTop: 10,
}}>Регистрация</OpenURLButton>
                        </ScrollView>
                    </SafeAreaView>
                </Content>
            </Container>

        );
    }
}

const windowWidth = Dimensions.get('window').width;
const windowHeight = Dimensions.get('window').height;
const data1 = [
    { key: 'Вейкборд :: МАЛЬЧИКИ' },
```

```
    { key: 'Вейкборд :: ДЕВОЧКИ' },
    { key: 'Вейкборд :: ЮНОШИ' },
    { key: 'Вейкборд :: ДЕВУШКИ' },
    { key: 'Вейкборд :: ЮНИОРЫ' },
    { key: 'Вейкборд :: ЮНИОРКИ' },
    { key: 'Вейкборд :: ЛЮБИТЕЛИ МУЖЧИНЫ' },
    { key: 'Вейкборд :: ЛЮБИТЕЛИ ЖЕНЩИНЫ' },
];
const regURL = "https://wakebase.ru/comp/145#registration";
const OpenURLButton = ({ url, children }) => {
    const handlePress = useCallback(async () => {
        const supported = await Linking.canOpenURL(url);
        if (supported) {
            await Linking.openURL(url);
        } else {
            Alert.alert(`Don't know how to open this URL: ${url}`);
        }
    }, [url]);

    return <Button title={children} onPress={handlePress} />;
};

const styles = StyleSheet.create({
    categoryTitleStyle: {
        padding: 10,
        fontSize: 18,
    },
    flatListContent:{
        padding: 5,
        paddingLeft: 40,
        fontSize: 18,
        height: 30,
    },
    nameCompetition:{
        textAlign: "center",
        paddingTop: 10,
        paddingBottom: 10,
        fontSize: 21
    },
    imageStyle:{
        width: windowWidth * 0.95,
        height: 300,
        marginLeft: "auto",
        marginRight: "auto",
        paddingBottom: 10,
    },
    textStyle:{
        padding: 10,
        fontSize: 18,
        height: 30,
    },
})
```

Листинг Г.17 – файл src/screens/privateScope/locationparks/ index.js

```
import { createStackNavigator } from 'react-navigation-stack';

import LocationParksScreen from './LocationParksScreen';

export default (LessonsNavigator = createStackNavigator(
  {
    Location: { screen: LocationParksScreen },
  },
  {
    initialRouteName: 'Location',
  }
));
```

Листинг Г.18 – файл LocationParkScreen.js

```
import React, { Component } from 'react';
import {
  Text,
  StyleSheet, Dimensions, View, Image, Alert,
} from 'react-native';
import { Drawer } from 'native-base';
import MapView, { Marker } from 'react-native-maps';
import MarkerIcon from './img/323.png';
import Icon from 'react-native-vector-icons/FontAwesome';
import Geolocation from '@react-native-community/geolocation';
import SideBar from "./SideBarPark";

import Api from '../../../lib/Api';

export default class LocationParksScreen extends Component {

  constructor(props) {
    super(props);

    this.state = {
      points: null,
      selectedPoint: null
    }
    this.updatePoint= this.updatePoint.bind(this)
  }

  componentDidMount() {
    this.loadData();
  }

  async loadData() {
    try {
      const response = await Api.get(`/points`);
```

```
      this.setState({
        points: response.data,
      });
    } catch (error) {
      console.error(error);
    }
  }

  updatePoint(point){
    this.setState({
      selectedPoint: point
    });
    //alert(this.state.selectedPoint && this.state.selectedPoint.id)
    this.openDrawer();
  }

  onMarkerPress(e) {
    this.props.points &&
      this.setState({
        selectedPoint: this.props.points.find(point => point.id ==
parseInt(e.nativeEvent.id))
      });
    {

      //alert(this.state.selectedPoint && this.state.selectedPoint.id)
    }
    this.openDrawer();
  }

  closeDrawer = () => {
    this.drawer._root.close()
  }

  openDrawer = () => {
    this.drawer._root.open()
  }

  render() {
    return (
      <Drawer
        ref={(ref) => { this.drawer = ref; }}
        side='bottom'
        openDrawerOffset={(viewport) => viewport.height - 410} // Отступ
сверху, 390 высота PointPreviewPanel
        panCloseMask={0.7} // Место с которого можно смахнуть панель
        content={
          <SideBar navigator={this.navigator}
            point={this.state.selectedPoint && this.state.selectedPoint} />}
        onClose={() => this.closeDrawer()}>

        <View style={styles.container}>
          <MapView style={styles.map}
            showsUserLocation
```

```
            initialRegion={{
              latitude: 53.124637,
              longitude: 45.062165,
              latitudeDelta: 0.8022,
              longitudeDelta: 0.8021,
            }}
          >
            {this.state.points && this.state.points.map(point => (
              <Marker draggable
                key={point.id}
                identifier={point.id.toString()}
                coordinate={{
                  latitude: parseFloat(point.latitude),
                  longitude: parseFloat(point.longitude)
                }}
                onPress={(e) => this.updatePoint(point)}
              >
                <Image
                  source={MarkerIcon}
                  style={{ width: 25, height: 40 }} />
              </Marker>
            )
            )}
          </MapView>
        </View>
      </Drawer>
    );
  }
}

let { height, width } = Dimensions.get('window')
const styles = StyleSheet.create({
  container: {
    height: height,
    width: width,
  },
  map: {
    ...StyleSheet.absoluteFillObject,
  }
});
```

Листинг Г.19 – файл SideBarPark.js

```
import PropTypes from 'prop-types';
import React, { Component } from 'react';
import { NavigationActions } from 'react-navigation';
import { ScrollView, Text, View, Dimensions, SafeAreaView, Image, FlatList,
StyleSheet } from 'react-native';
import { StackNavigator } from 'react-navigation';
import { Linking } from 'react-native';
import Api from '../../../lib/Api';
```

```
class SideMenu extends Component {

        state = {
            point: PropTypes.object
        }

    navigateToScreen = (route) => () => {
        const navigateAction = NavigationActions.navigate({
            routeName: route
        });
        this.props.navigation.dispatch(navigateAction);
    }

    render() {
        return (

            <SafeAreaView>
                <View style={styles.viewMain}>
                    <ScrollView
                        contentInsetAdjustmentBehavior="automatic"
                        style={{}}>
                        <Text style={styles.titleStyle}>
                            {this.props.point &&
this.props.point.name}</Text>
                        <Text style={styles.textStyle}>
        {this.props.point && this.props.point.addres}</Text>
        {this.props.point && <Image source={{uri: this.props.point.image}}
//{require("./img/barbados.jpg")}  //{{uri: this.props.point &&
this.props.point.image}}//
                            style={styles.imageStyle} />}
                        <Text style={styles.textStyle}
                            onPress={() => { Linking.openURL(this.props.point
&& this.props.point.phone_use_url); }}>
                            Контактный телефон: {this.props.point &&
this.props.point.phone_number}</Text>
                        <Text style={styles.textStyle}>Цена:</Text>
                        <Text style={styles.listItemStyle}>{this.props.point
&& this.props.point.one_set}</Text>
                        <Text style={styles.listItemStyle}>{this.props.point
&& this.props.point.all_day}</Text>
                    </ScrollView>
                </View>
            </SafeAreaView>
        );
    }
}

SideMenu.propTypes = {
    navigation: PropTypes.object
};

export default SideMenu;
```

```
let { height, width } = Dimensions.get('window')
//let phone=this.props.point && this.props.point.phoneNumber;
const styles = StyleSheet.create({
    viewMain:
    {
        backgroundColor: "white",
        height: height,
        borderTopRightRadius: 25,
        borderTopLeftRadius: 25,
    }
    ,
    titleStyle: {
        textAlign: "center",
        fontSize: 24,
        paddingTop: 5,
    },
    textStyle: {
        paddingTop: 5,
        fontSize: 18,
        paddingLeft: 20
    },
    imageStyle: {
        marginTop: 5,
        width: width * 0.9,
        height: height * 0.2,
        marginLeft: "auto",
        marginRight: "auto",
    },
    listItemStyle: {
        paddingTop: 5,
        fontSize: 18,
        paddingLeft: 30
    }

})
```

Приложение Д

СКРИНШОТЫ ЗАПРОСОВ

(рекомендуемое)

Рисунок Д.1 – Получение списка уроков

Рисунок Д.2 – Получение данных урока номер 2

Рисунок Д.3 – Получение списка маркеров

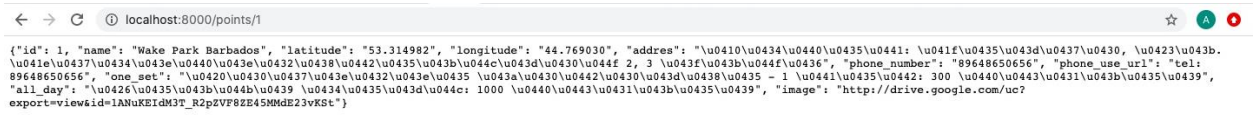


Рисунок Д.4 – Получение данных маркера номер 1

Рисунок Д.5 – Получение списка соревнований

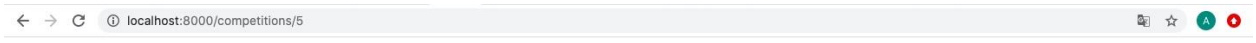


Рисунок Д.6 – Получение данных о соревнованиях номер 5

Рисунок Д.7 – Получение списка вопросов

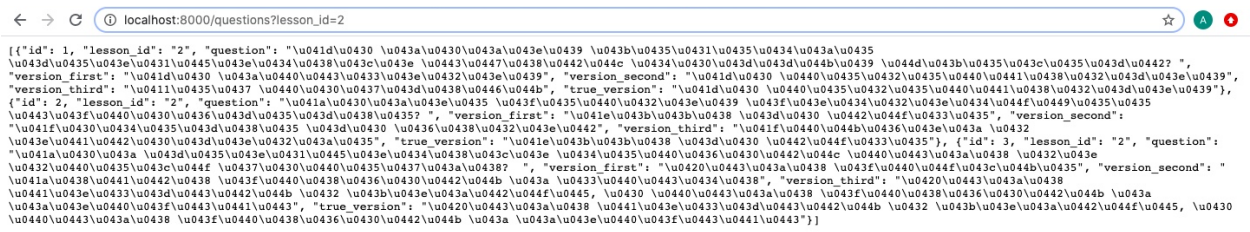


Рисунок Д.8 – Получение списка вопросов, относящихся к 7 уроку

Приложение Е.

СКРИНШОТЫ ЮЗАБИЛИТИ-ТЕСТИРОВАНИЯ МОБИЛЬНОГО ПРИЛОЖЕНИЯ

(рекомендуемое)

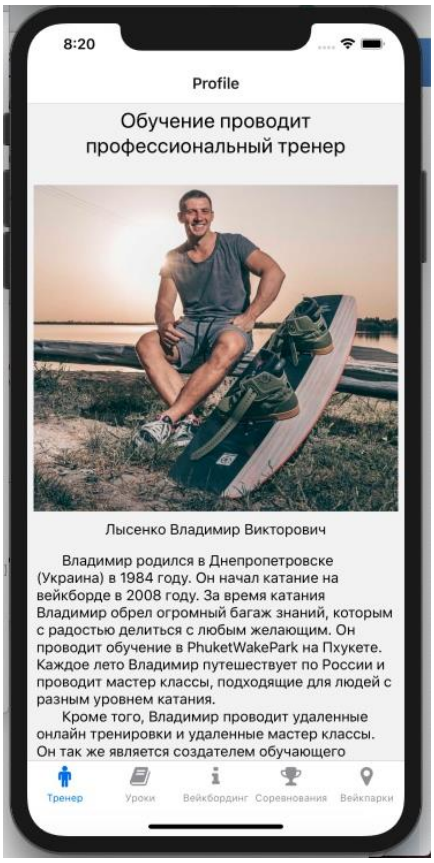


Рисунок Е.1а – Отображение страницы «Тренер» на IOS

Рисунок Е.1б – Отображение страницы «Тренер» на IOS

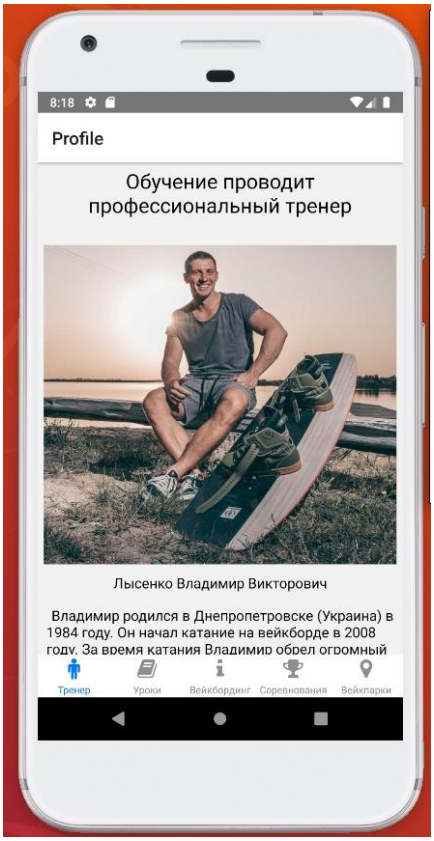


Рисунок Е.2а – Отображение страницы «Тренер» на Android

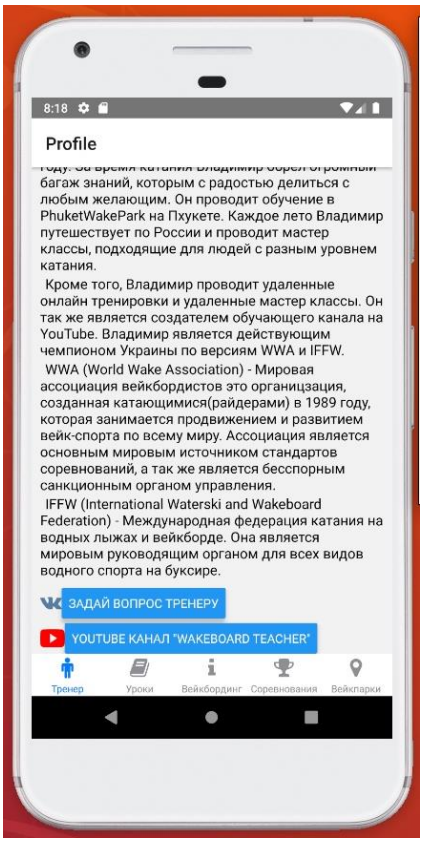


Рисунок Е.2б – Отображение страницы «Тренер» на Android

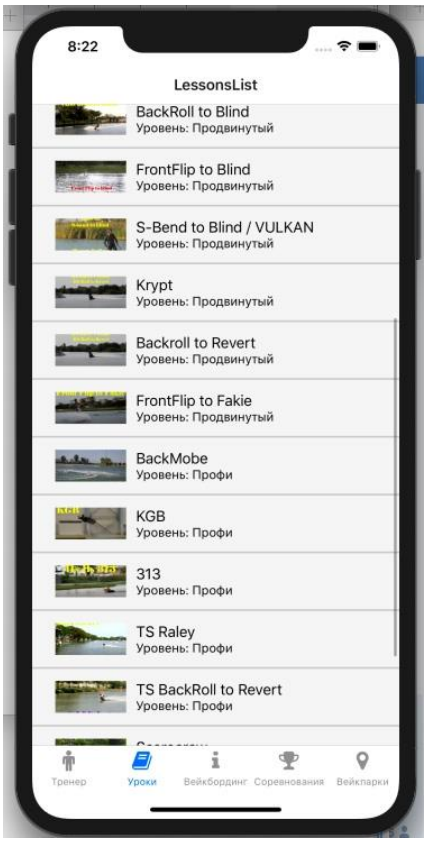


Рисунок Е.3 – Отображение страницы «Уроки» на IOS

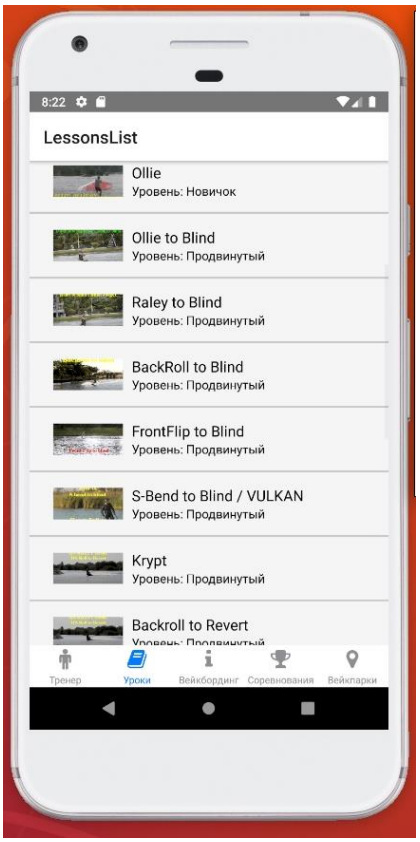


Рисунок Е.4 – Отображение страницы «Уроки» на Android

Рисунок Е.5 – Отображение страницы выбранного урока на IOS

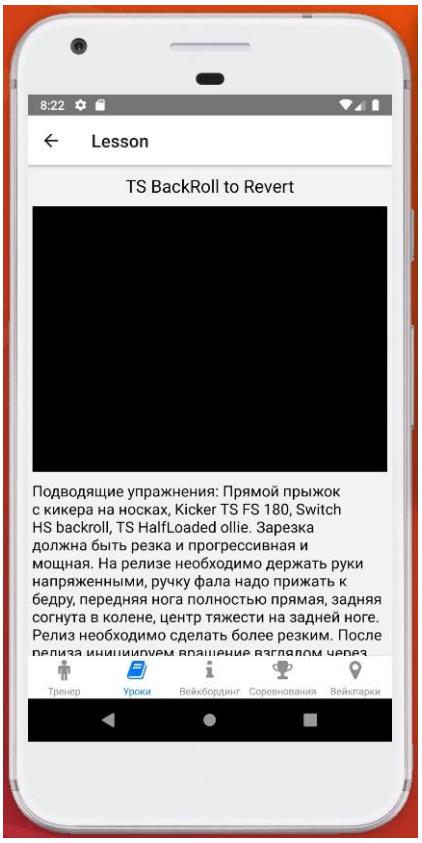


Рисунок Е.6 – Отображение страницы выбранного урока на Android

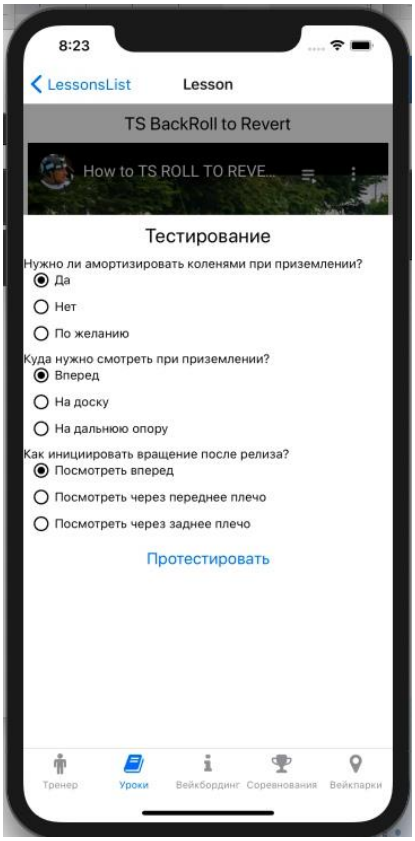


Рисунок Е.7– Отображение страницы «Тестирование» на IOS

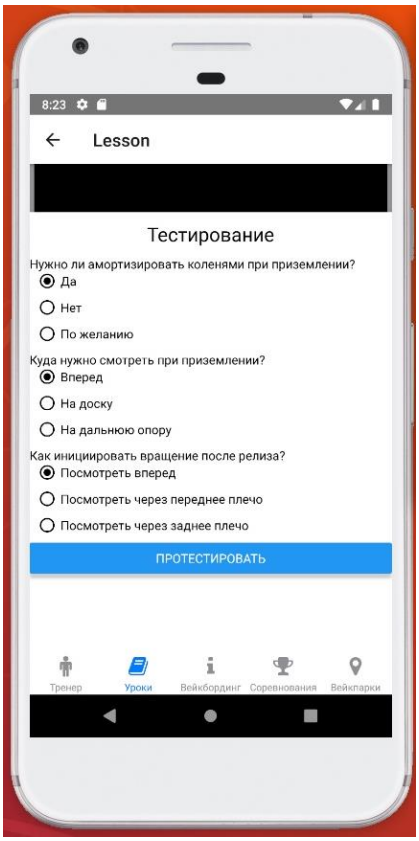


Рисунок Е.8 – Отображение страницы «Тестирование» на Android

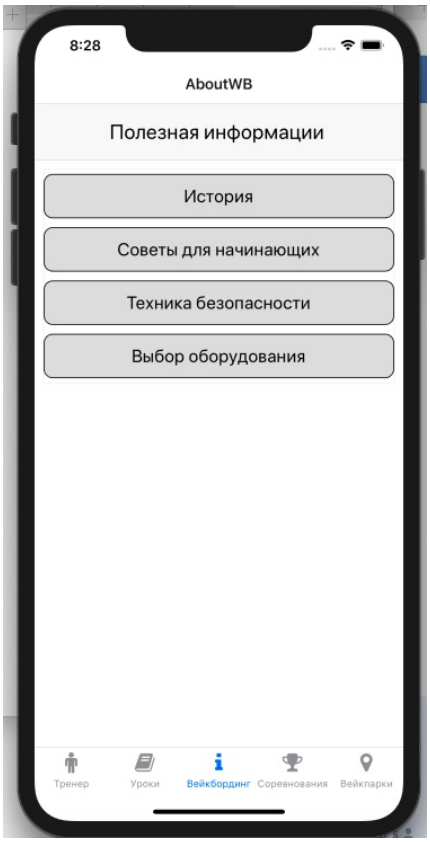


Рисунок Е.9 – Отображение страницы «Вейкбординг» на IOS

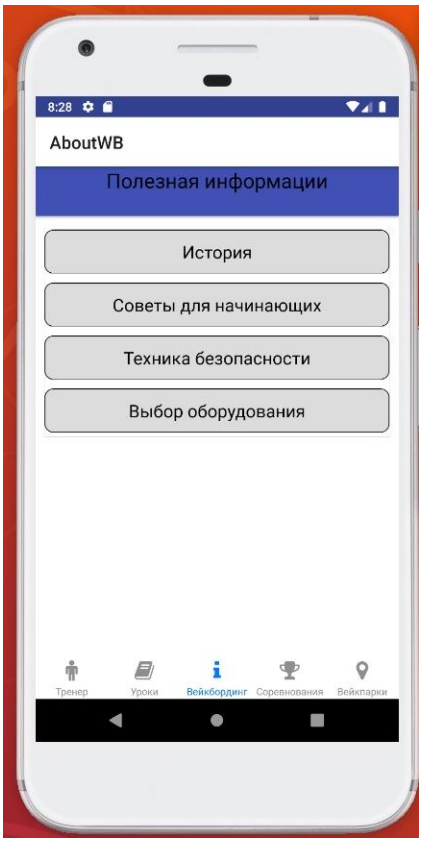


Рисунок Е.10 – Отображение страницы «Вейкбординг» на Android

Рисунок Е.11 – Отображение информации об истории вейкбординга на IOS

Рисунок Е.12 – Отображение информации об истории вейкбординга на Android

Рисунок Е.13 – Отображение советов для начинающих на IOS

Рисунок Е.14 – Отображение советов для начинающих на Android

Рисунок Е.15 – Отображение техники безопасности на IOS

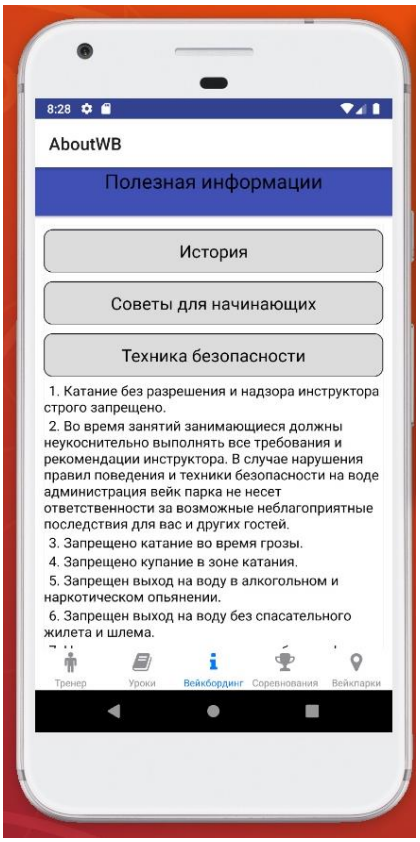


Рисунок Е.16 – Отображение техники безопасности на Android

Рисунок Е.17 – Отображение информации о выборе оборудования на IOS

Рисунок Е.18 – Отображение информации о выборе оборудования на Android

Рисунок Е.19 – Отображение страницы «Соревнования» на IOS

Рисунок Е.20 – Отображение страницы «Соревнования» на Android

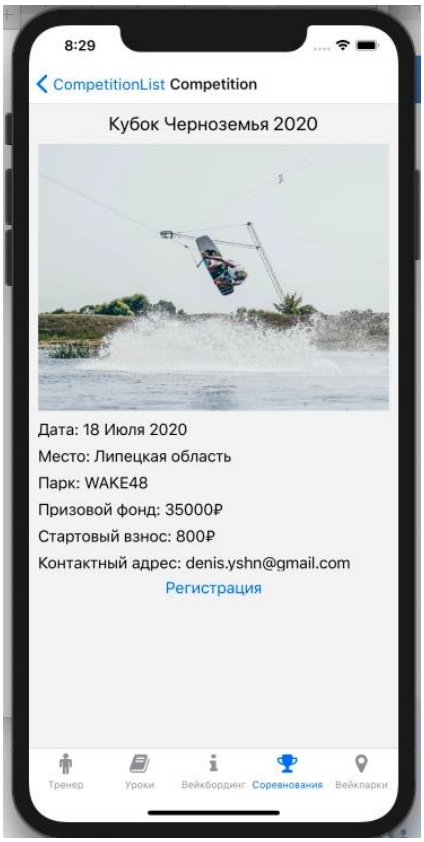


Рисунок Е.21 – Отображение данных о соревнованиях на IOS

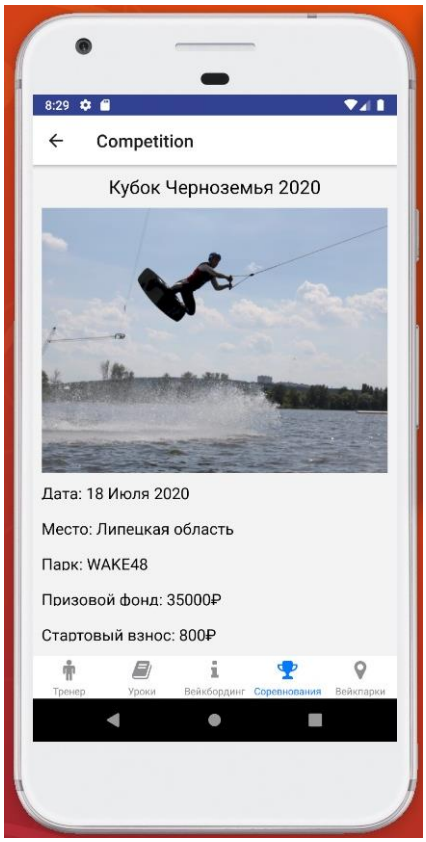


Рисунок Е.22 – Отображение данных о соревнованиях на Android

Рисунок Е.23 – Отображение карты и маркеров на IOS

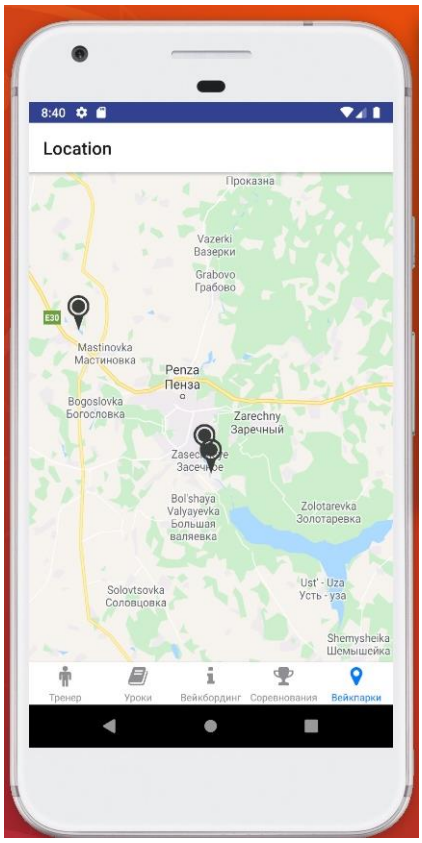


Рисунок Е.24 – Отображение карты и маркеров на Android

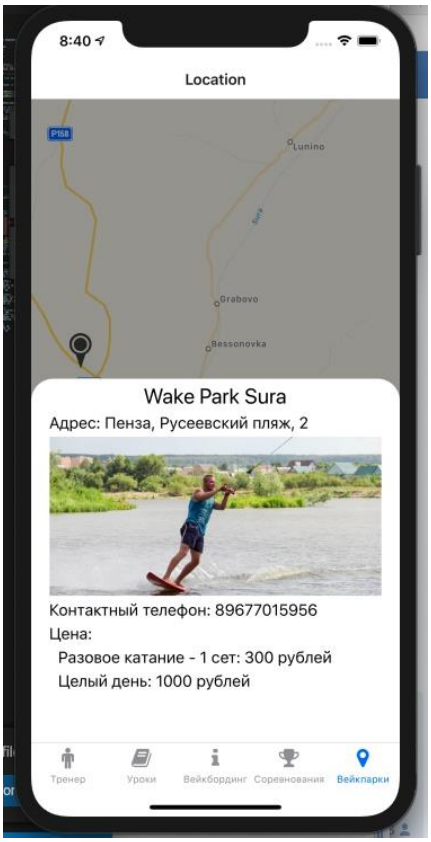


Рисунок Е.25 – Отображение информации о вейк станции на IOS

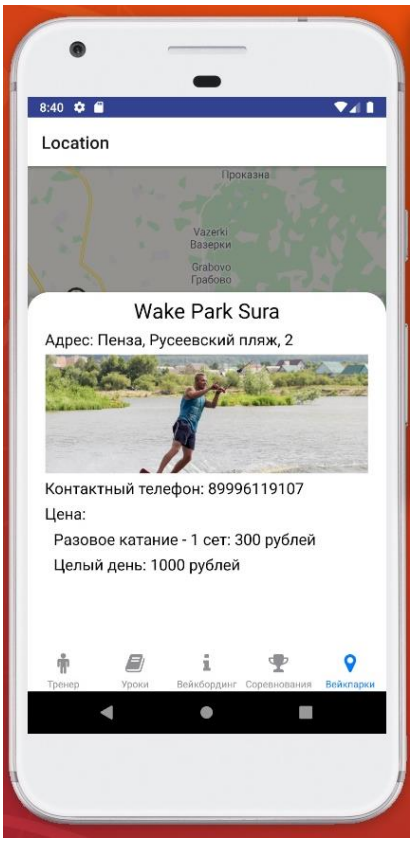


Рисунок Е.26 – Отображение информации о вейк станции на Android

Приложение Ж
РЕЗУЛЬТАТЫ ФУНКЦИОНАЛЬНОГО ТЕСТИРОВАНИЯ МОБИЛЬНОГО ПРИЛОЖЕНИЯ
(рекомендуемое)

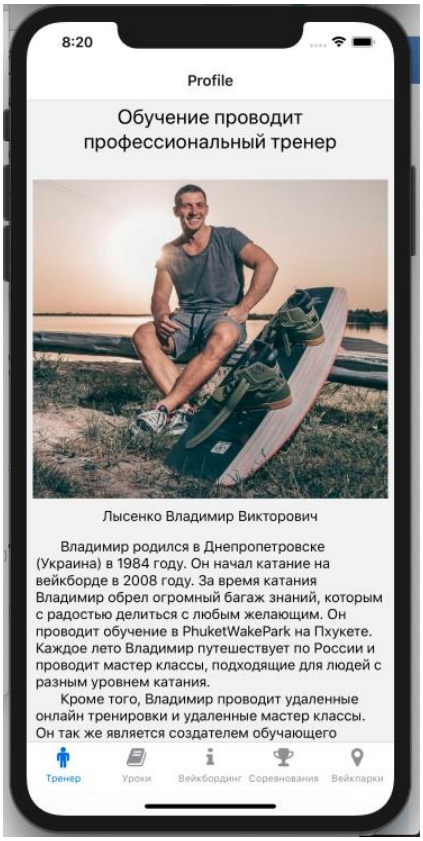


Рисунок Ж.1а – Отображение страницы «Тренер» на IOS

Рисунок Ж.1б – Отображение страницы «Тренер» на IOS

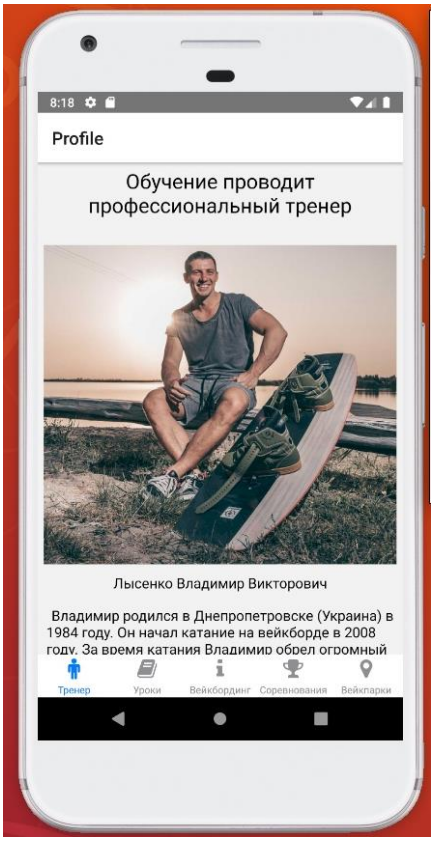


Рисунок Ж.2а – Отображение страницы «Тренер» на Android

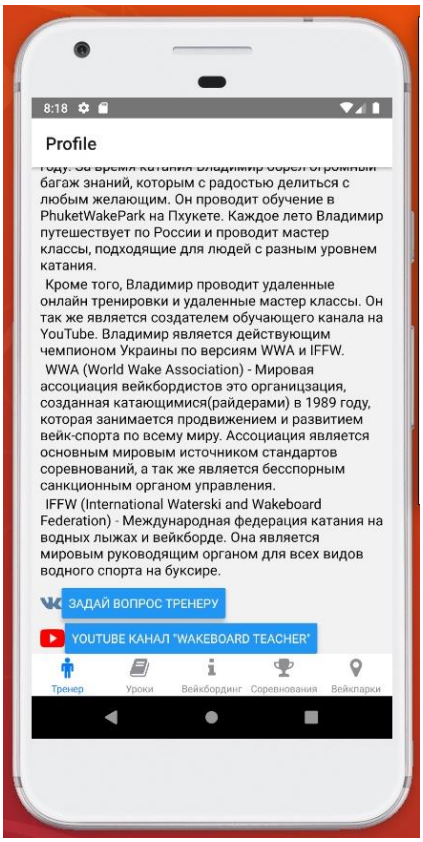


Рисунок Ж.2б – Отображение страницы «Тренер» на Android

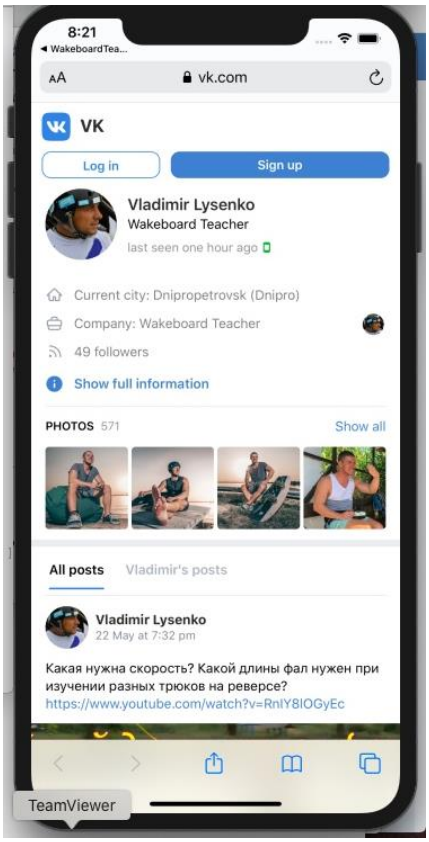


Рисунок Ж.3 – Отображение страницы тренера в соц. Сети VK на IOS

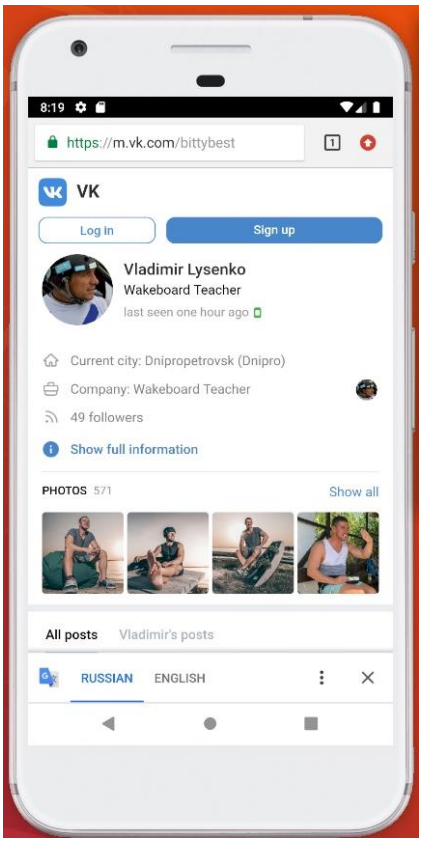


Рисунок Ж.4 – Отображение страницы тренера в соц. Сети VK на Android

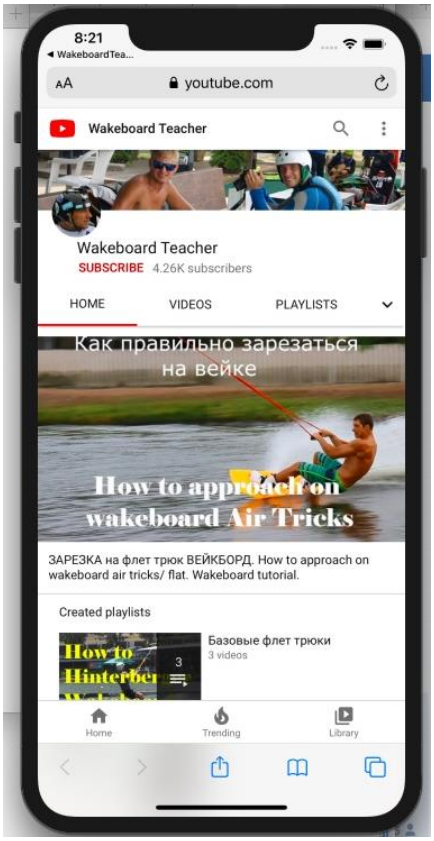


Рисунок Ж.5 – Отображение страницы youtube канала на IOS

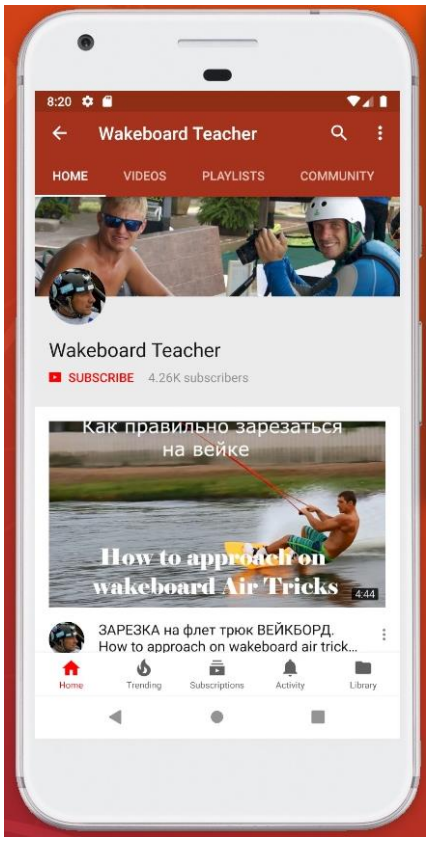


Рисунок Ж.6 – Отображение страницы youtube канала на Android

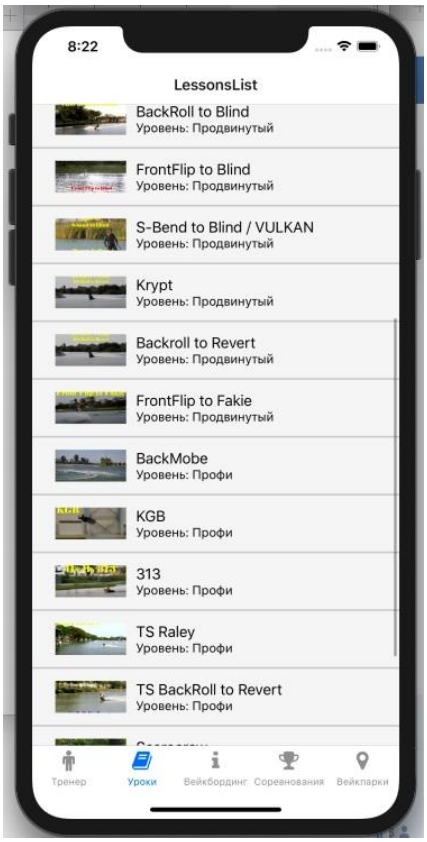


Рисунок Ж.7 – Отображение списка уроков на IOS

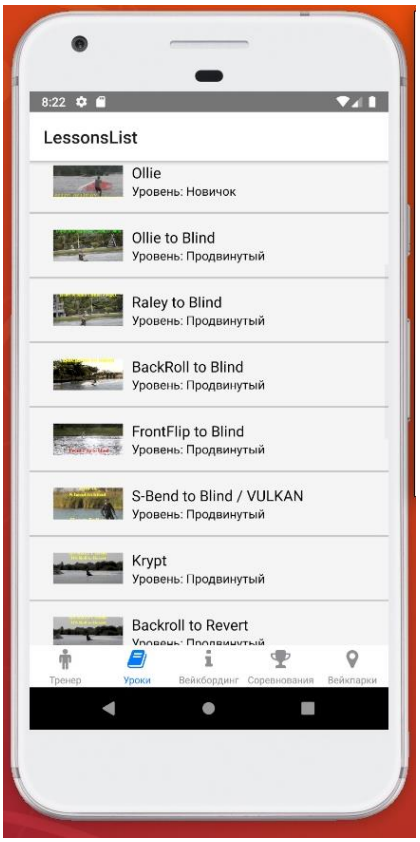


Рисунок Ж.8 – Отображение списка уроков на Android

Рисунок Ж.9 – Отображение страницы выбранного урока на IOS

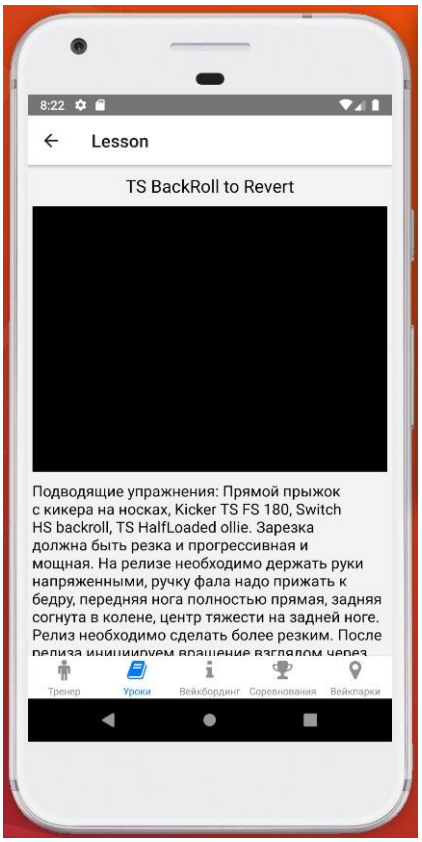


Рисунок Ж.10 – Отображение страницы выбранного урока на Android

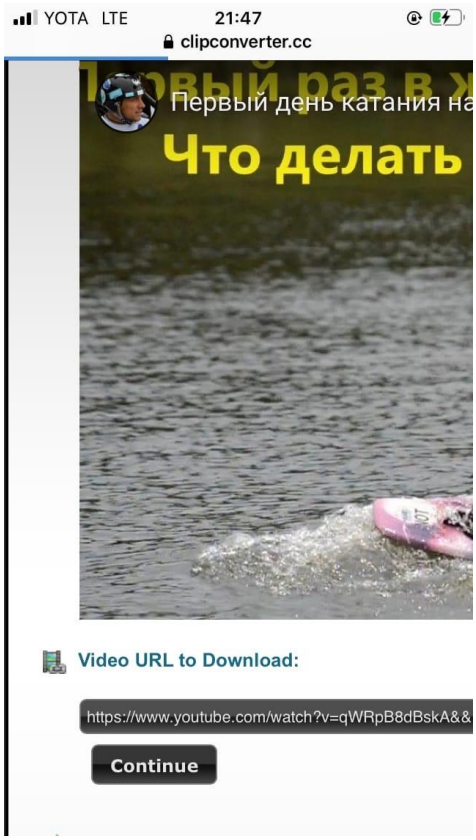


Рисунок Ж.11 – Отображениестраницы веб-сервиса на IOS

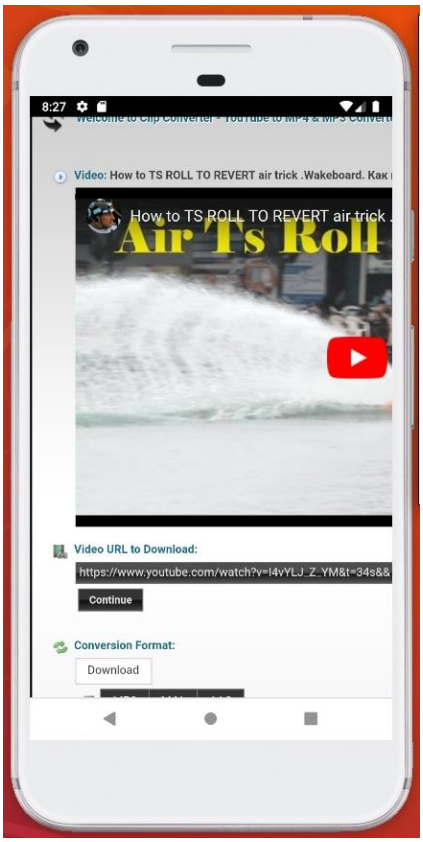


Рисунок Ж.12 – Отображение страницы веб-сервиса на Android

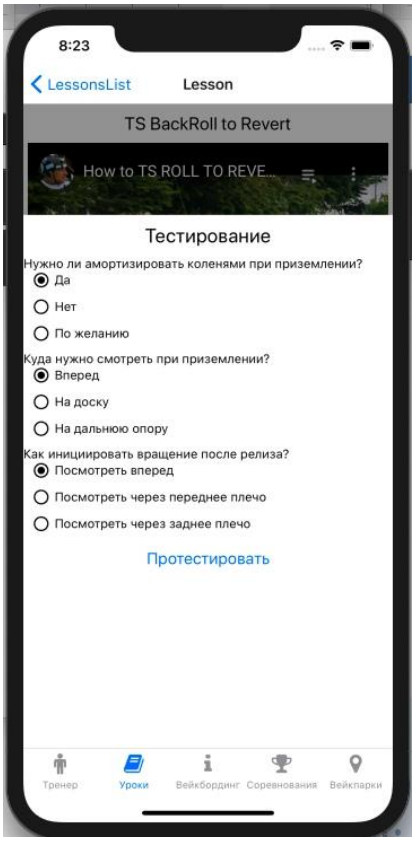


Рисунок Ж.13– Отображение страницы «Тестирование» на IOS

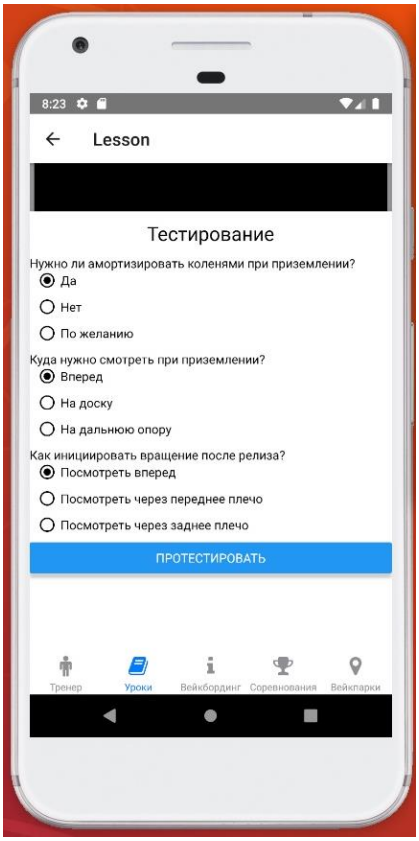


Рисунок Ж.14 – Отображение страницы «Тестирование» на Android

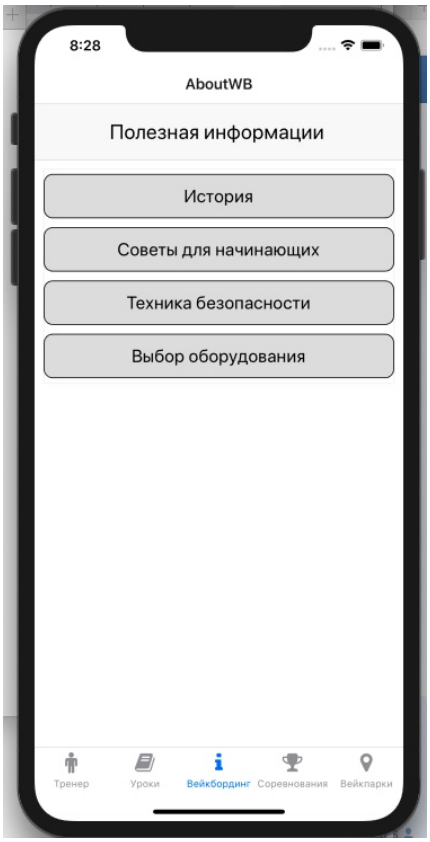


Рисунок Ж.15 – Отображение страницы «Вейкбординг» на IOS

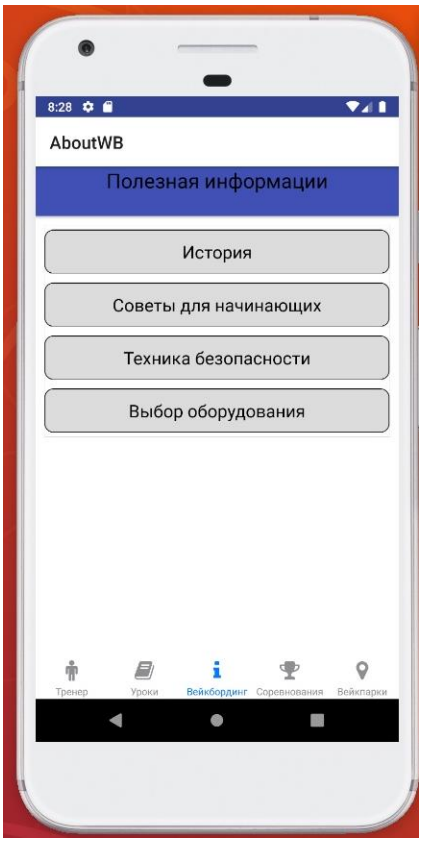


Рисунок Ж.16 – Отображение страницы «Вейкбординг» на Android

Рисунок Ж.17 – Отображение информации об истории вейкбординга на IOS

Рисунок Ж.18 – Отображение информации об истории вейкбординга на Android

Рисунок Ж.19 – Отображение советов для начинающих на IOS

Рисунок Ж.20 – Отображение советов для начинающих на Android

Рисунок Ж.21 – Отображение техники безопасности на IOS

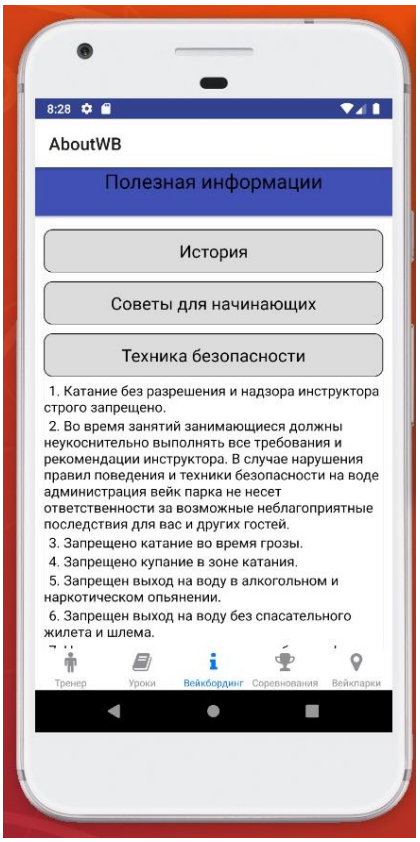


Рисунок Ж.22 – Отображение техники безопасности на Android

Рисунок Ж.23 – Отображение информации о выборе оборудования на IOS

Рисунок Ж.24 – Отображение информации о выборе оборудования на Android

Рисунок Ж.25 – Отображение списка соревнований на IOS

Рисунок Ж.26 – Отображение списка соревнований на Android

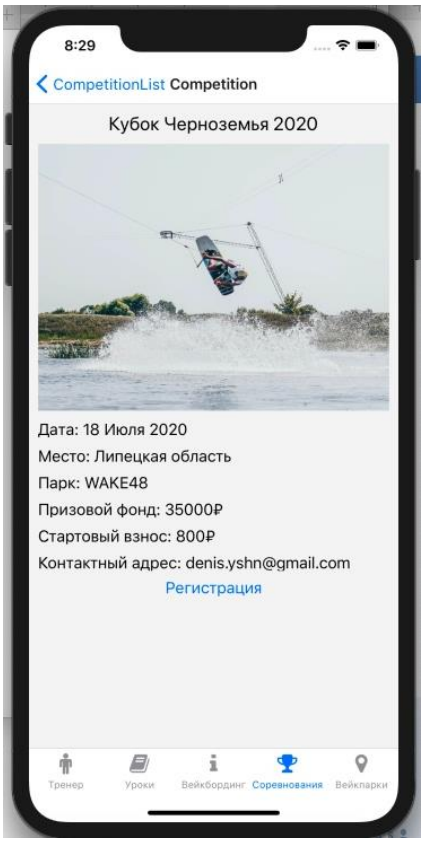


Рисунок Ж.27 – Отображение данных о соревнованиях на IOS

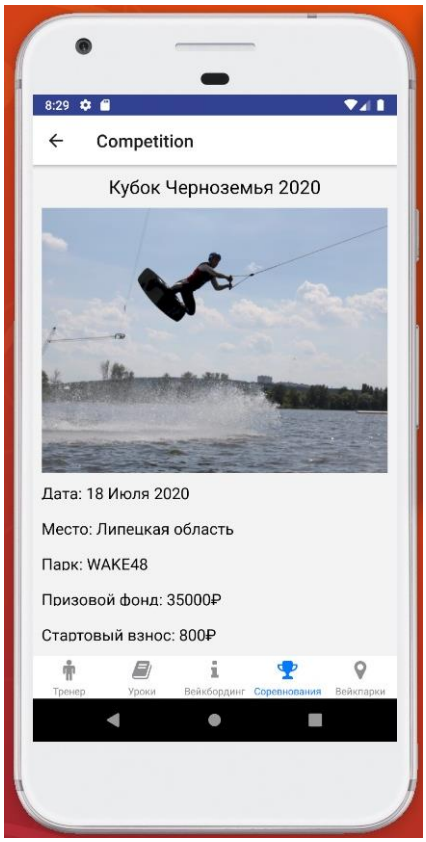


Рисунок Ж.28 – Отображение данных о соревнованиях на Android

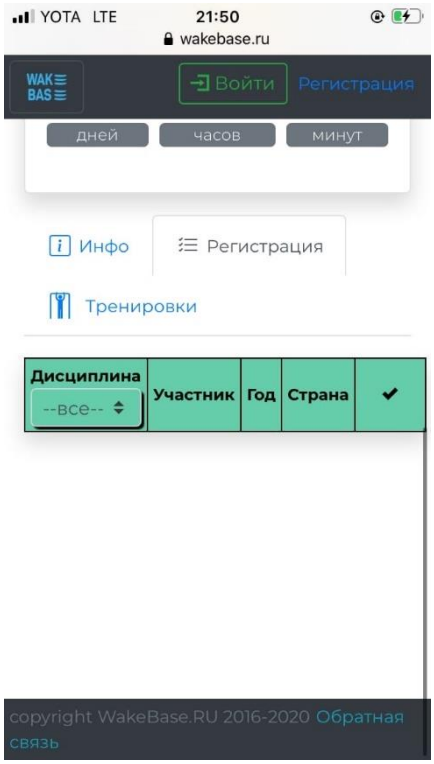


Рисунок Ж.29 – Отображение страницы регистрации на сайте wakebase.ru на на IOS

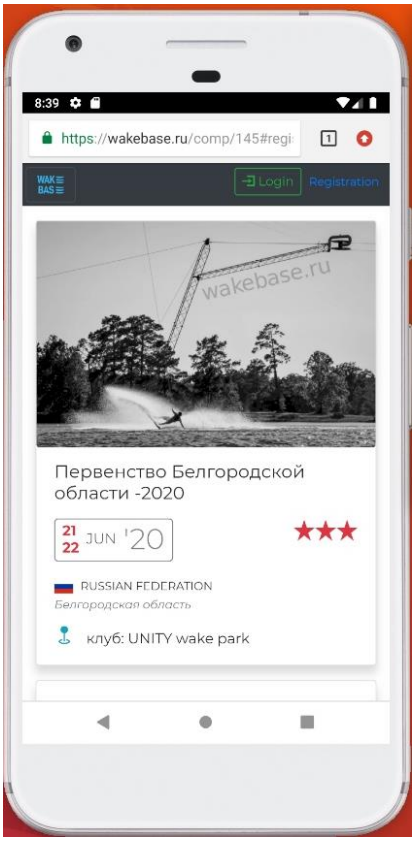


Рисунок Ж.30 – Отображение страницы регистрации на сайте wakebase.ru на Android

Рисунок Ж.31 – Отображение карты и маркеров на IOS

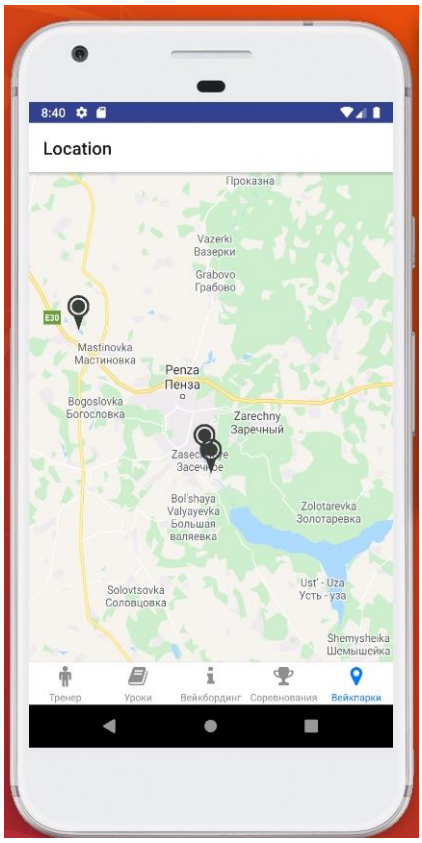


Рисунок Ж.32 – Отображение карты и маркеров на Android

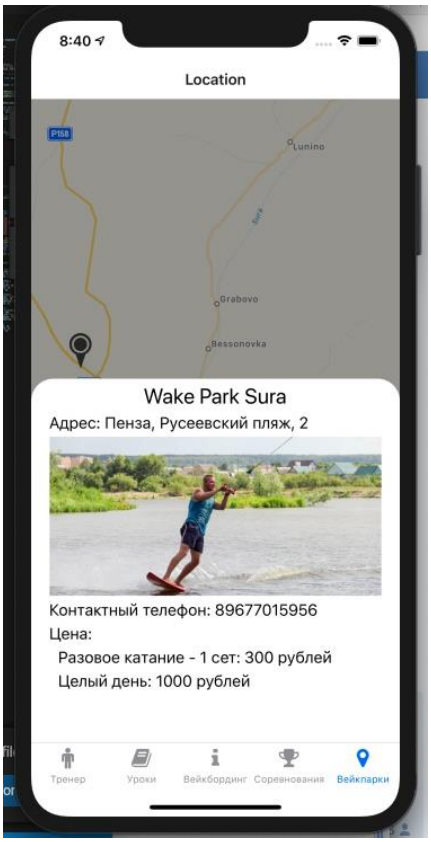


Рисунок Ж.33 – Отображение информации о вейк станции на IOS

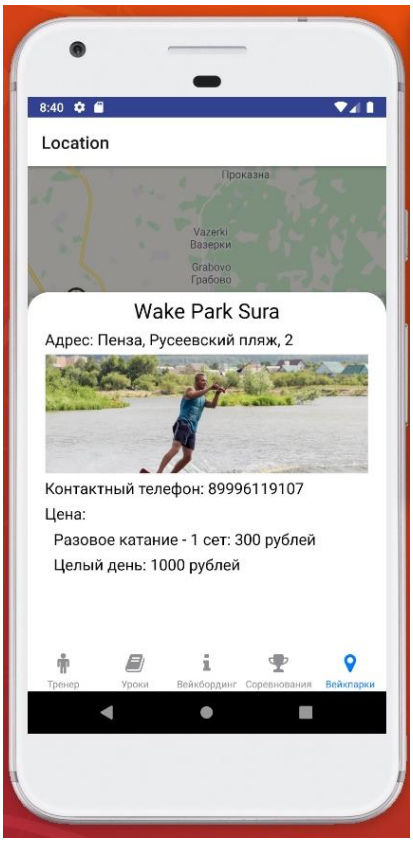


Рисунок Ж.34 – Отображение информации о вейк станции на Android

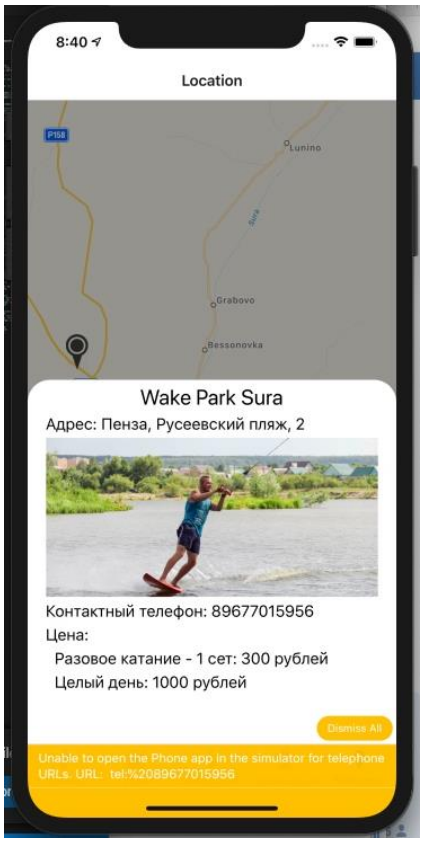


Рисунок Ж.35 – Отображение предупреждения фреймворка на IOS

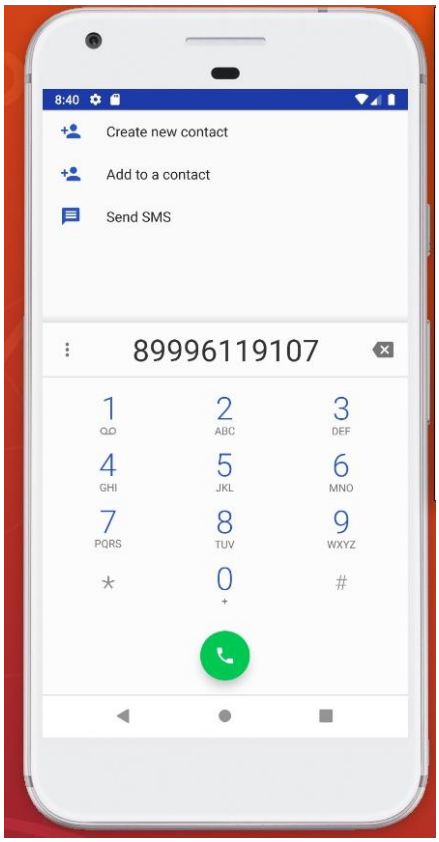


Рисунок Ж.36 – Отображение приложения «Телефон» на Android