

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 поддержки процессов
взаимного поиска потребителей и
поставщиков социальных услуг

Студент

 18.06.21 Ермин Егор Александрович
(подпись, дата) (ФИО полностью)

Руководитель

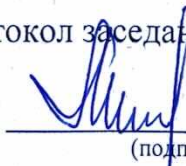
 18.06.21 Князев В.Н.
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр

 18.06.21 Такташкин Д.В.
(подпись, дата) (фамилия, инициалы)

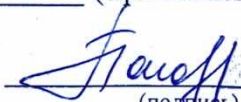
Работа допущена к защите (протокол заседания кафедры от _____ № _____)

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, 2021

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

Факультет
Вычислительной техники

Кафедра
МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
П.П. Макарычев
«22» _____ 2020 г

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА

1. Студент Ермин Егор Александрович гр. 17ВП1 факультета ВТ
<Фамилия, имя, отчество полностью>

2. Тема работы Мобильное приложение поддержки процессов взаимного поиска
потребителей и поставщиков социальных
услуг

Тема утверждена приказом ПГУ № 0017-д/о от " 28 " декабря 2020

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 мобильное приложение поддержки процессов взаимного поиска
потребителей и поставщиков социальных услуг

2) Среда выполнения: Операционная система iOS 9.0 и выше для клиентской части,
операционная система Linux для серверной части

Языковые средства разработки: Swift, SQL, UML, PHP

Инструментальные средства разработки: Xcode 12.4, смартфон iPhone SE (2nd
generation), Microsoft Project, Microsoft Visio, Draw.io, MySQL Server

Технологии разработки: RUP, ООП, шаблон MVC, Web API

3) Основные функции мобильного приложения:

а) просмотр каталога исполнителей по типу услуги;

б) создание тендеров;

в) добавление услуг в избранное;

г) просмотр конкурсных подборок;

д) создание собственных услуг.

5. Перечень подлежащих разработке вопросов

1) Анализ предметной области и постановка задач

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 %	14.04.2021 - 22.04.2021	
2	Проектирование	25 %	23.04.2021 - 06.05.2021	
3	Реализация	40 %	07.05.2021 - 31.05.2021	
4	Контроль качества программного обеспечения	5 %	01.05.2021 - 03.06.2021	
5	Оформление пояснительной записки	15 %	04.06.2021 - 14.06.2021	

Дата выдачи задания "28" декабря 2020

Руководитель бакалаврской работы



28.12.2020

(подпись, дата)

Задание к исполнению принял студент



28.12.2020

(подпись, дата)

Работу к защите допустить



28.12.2020.

Декан факультета

(подпись, дата)

Реферат

Пояснительная записка содержит 144 листа, 79 рисунков, 8 таблиц, 24 использованных источника и 6 приложений.

МОБИЛЬНОЕ ПРИЛОЖЕНИЕ, iOS, IDEF1X, БАЗА ДАННЫХ, MVC, REST API, SWIFT, XCODE, PHP, SQL

Объектом разработки является мобильное приложение поддержки процессов взаимного поиска потребителей и поставщиков социальных услуг.

Цель выпускной квалификационной работы – разработать мобильное приложения поддержки процессов взаимного поиска потребителей и поставщиков социальных услуг.

В результате выполненной работы создано мобильное приложение поддержки процессов взаимного поиска потребителей и поставщиков социальных услуг, позволяющее найти интересующую услугу по её типу, а также выложить собственную.

Разработка проводилась на языке программирования Swift в среде разработки Xcode 12.4. Среда исполнения мобильного приложения iOS 9.0 и выше. В качестве СУБД была выбрана MySQL Server. Тестирование серверной части приложения проводилось с помощью инструментального средства Postman.

					ПГУ 09.03.04– 8BP171.09 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата	«Мобильное приложение поддержки процессов взаимного поиска потребителей и поставщиков социальных услуг» Пояснительная записка	Лит.	Лист	Листов
Разраб.	Ермин Е.А.			18.06.21				
Провер.	Князев В.Н.			18.06.21			3	144
Н. Контр.	Такташкин Д.В.			21.06.21		Группа 17ВП1		
Утверд.	Макарычев П.П.			22.06				

Содержание

Введение.....	6
1 Анализ предметной области и требований.....	8
1.1 Анализ предметной области и постановка задачи.....	8
1.2 Анализ аналогичных решений на рынке мобильных приложений	9
1.2.1 Анализ мобильного приложения «Свадебный эксперт»	9
1.2.2 Анализ мобильного приложения «МойПрофи»	10
1.2.3 Анализ мобильных приложений «Яндекс. Услуги».....	11
1.3 Анализ функциональных требований	13
1.4 Анализ нефункциональных требований	17
1.5 Построение исходной концептуальной модели предметной области	18
2 Проектирование мобильного приложения	22
2.1 Выбор технологий и инструментальных средств разработки	22
2.2 Планирование разработки мобильного приложения и оценка бюджета ..	22
2.3 Проектирование пользовательского интерфейса.....	26
2.4 Проектирование архитектуры приложения.....	45
2.5 Проектирование серверной части приложения	45
2.5.1 Логическое проектирование базы данных.....	45
2.5.2 Проектирование серверной части приложения	46
2.6 Проектирование клиентской части приложения	47
3 Реализация мобильного приложения	49
3.1 Реализация серверной части мобильного приложения	49
3.1.1 Реализация базы данных.....	49
3.1.2 Реализация серверной части	50
3.2 Реализация клиентской части мобильного приложения.....	50
4 Тестирование программного продукта	53
4.1 Тестирование серверной части	53
4.2 Тестирование клиентской части	54
4.2.1 Функциональное тестирование.....	54
4.2.2 Модульное тестирование.....	58

4.2.3 Метрики кода	59
Заключение	63
Список использованных источников	64
Приложение А. Глоссарий	67
Приложение Б. Исходный код серверной части	69
Приложение В. Исходный код клиентской части.....	81
Приложение Г. Результаты тестирования серверной части	126
Приложение Д. Результаты тестирования клиентской части.....	132
Приложение Е. Акт о внедрении	143

Введение

Информационные технологии все более глубоко внедряются во все области нашей жизни. На сегодняшний день сфера IT является одной из наиболее развивающихся отраслей в мире.

Одним из направлений государственной политики Российской Федерации является развитие информационных технологий. 9 мая 2017 г. была утверждена Стратегия развития информационного общества в Российской Федерации на 2017 – 2030 годы [1]. В данной стратегии рассматриваются цели и задачи политики Российской Федерации по развитию отрасли информационных технологий [1].

А 28 июля 2017 г. утверждена Программа «Цифровая экономика Российской Федерации» [2]. Программа направлена на создание благоприятных условий для развития цифровой экономики Российской Федерации, что поспособствует увеличению конкурентоспособности страны, экономическому росту, а также приведет к улучшению качества жизни граждан.

В настоящее время одной из проблем современного мира является совершенствования социальной системы. Эта проблема актуальна, поскольку без эффективного функционирования этой системы стабильность и прогресс не имеют места в социальной среде.

Применение информационных технологий в социальной сфере – это наиболее эффективное решение этой проблемы. Эти технологии обладают эластичностью, ликвидностью и приспособляемостью к воздействиям внешней среды. В ходе модернизации социальной среды информационные технологии представляют собой основной фундамент общества, так как с их помощью можно улучшить и автоматизировать процессы обработки данных информации, которые в последние годы занимают значимое место в жизненном процессе человеческого общества.

Настало такое время, когда человек без современных технологий уже не может обойтись ни дня и даже просто выйти из дома, так как в повседневной жизни люди получают большое количество информации за один день, которую записывают и носят с собой в телефонах, ноутбуках и другой компактной технике [3].

Особую важность представляет собой взаимный поиск поставщиков и потребителей социальных услуг. Информационные технологии позволяют ускорить этот процесс в несколько раз. Имея под рукой мобильный телефон с интересующими контактами, отпадает необходимость обзванивать кучу номеров в поиске нужной услуги и соответствующего специалиста.

Таким образом, можно с уверенностью заявить, что разработка мобильного приложения для поддержки взаимодействия поставщиков и потребителей социальных услуг очень актуальна в современном мире.

Целью данной работы является разработка мобильного приложения поддержки процессов взаимного поиска потребителей и поставщиков социальных услуг.

Для данной работы были выделены следующие задачи:

- анализ предметной области и требований;
- анализ аналогичных решений;
- проектирование серверной части приложения;
- проектирование клиентской части приложения;
- разработка серверной части приложения;
- разработка клиентской части приложения;
- тестирование и отладка.

1 Анализ предметной области и требований

1.1 Анализ предметной области и постановка задачи

Разрабатываемое мобильное приложение предназначено для взаимного поиска потребителей и исполнителей различного рода социальных услуг.

Приложение предоставляет пользователю возможность поиска нужной ему услуги с возможностью фильтрации по стоимости и типу, а также размещение своих собственных. На услугу можно оставить отзыв или просмотреть оставленные ранее отзывы, добавить в избранное, поделиться ей.

При размещении собственной услуги в обязательном порядке нужно указать её тип, отображаемое название, фотографии и стоимость.

Для заказчиков предусмотрена возможность создания тендера на услугу. Тендер – конкурентная форма отбора предложений на оплачиваемую поставку товаров, оказание услуг или выполнение работ по заранее объявленным условиям и в оговоренные сроки. После объявления заказчиком тендера, исполнитель с соответствующей услугой сможет рассмотреть предложение в соответствующем разделе приложения и связаться с клиентом.

Таким образом, пользователем мобильного приложения является лицо, заинтересованное в получении услуги, либо предоставляющее свою. Можно выделить следующие основные задачи, решаемые в данной предметной области:

- создание услуги с соответствующим типом;
- просмотр услуг по типу с возможностью фильтрации;
- предоставление возможности создания тендера для заказчика услуги.

Разрабатываемое приложение должно быть удобным в использовании поддерживать актуальные версии операционной системы, на которой будет функционировать.

1.2 Анализ аналогичных решений на рынке мобильных приложений

В настоящее время существуют частично аналогичные приложения, позволяющие поставщикам и потребителям социальных услуг взаимодействовать друг с другом. Проведем сравнительный анализ таких программных продуктов данного сектора, как «Свадебный эксперт», «МойПрофи» и «Яндекс. Услуги».

1.2.1 Анализ мобильного приложения «Свадебный эксперт»

Приложение «Свадебный эксперт» предоставляет возможность подготовиться к любому торжеству и празднику, а в первую очередь, к свадьбе. Главный экран мобильного приложения на платформе iOS представлен на рисунке 1 [4].

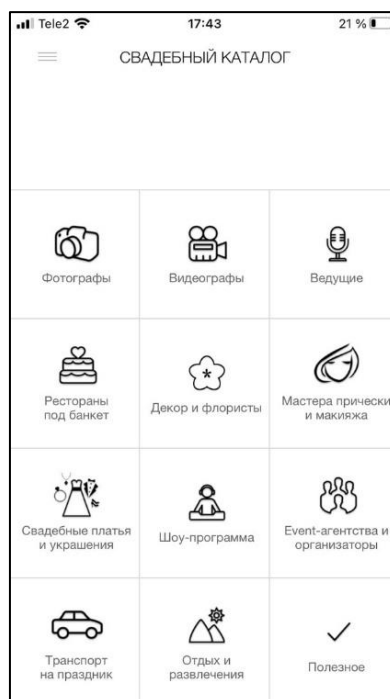


Рисунок 1 – Главный экран мобильного приложения «Свадебный эксперт»

Среди достоинств данного приложения можно выделить приятный дизайн и интуитивно понятный интерфейс. Приложение работает с городами большинства пост советских стран и содержит достаточно большую базу

исполнителей услуг, которых можно отсортировать по рейтингу. Среди достоинств так же можно выделить функцию органайзер, с помощью которой можно удобно следить за отсчётом времени до наступления торжества, создавать списки гостей и задач, и считать затраты на предстоящий праздник.

Недостатками данного приложения является отсутствие возможности поиска исполнителей и площадок/ресторанов по названию, а также ориентированность контента по большей части на свадьбы.

1.2.2 Анализ мобильного приложения «МойПрофи»

«МойПрофи» – приложение, позволяющее производить онлайн запись на услуги к мастерам красоты вашего города на прямую. Преимуществом данного приложения является возможность консультации по услугам через чат, а также получение эксклюзивных предложений на услуги от мастеров. Главный экран приложения представлен на рисунке 2 [5].

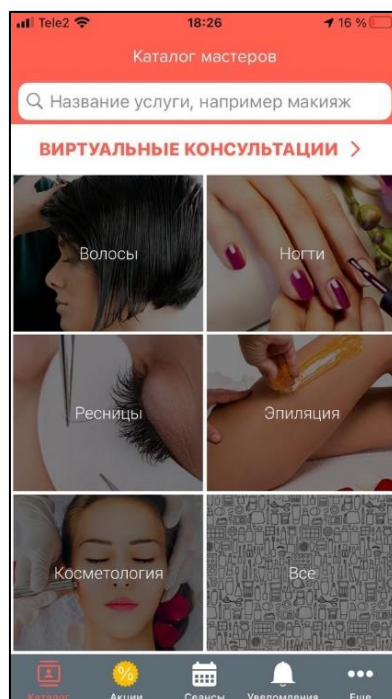


Рисунок 2 – Главный экран мобильного приложения «МойПрофи»

Удобной чертой данного приложения являются напоминания о запланированных сеансах.

К недостаткам можно отнести то, что приложение включает в себя функционал только для поиска мастеров. Если существует необходимость

разместить свою услугу – придётся скачивать версию приложения для исполнителей, функционал которого частично платный. Также к недостаткам можно отнести то, что логотип и название приложения не отражают его назначения. Создаётся ощущение, что помимо услуг в сфере красоты внутри него можно найти больше услуг.

1.2.3 Анализ мобильных приложений «Яндекс. Услуги»

В мобильном приложении «Яндекс. Услуги – уборка, ремонт» можно разместить заказ и выбрать специалиста по профилю, отзывам, рейтингу или стоимости услуги [6]. Главный экран приложения представлен на рисунке 3.

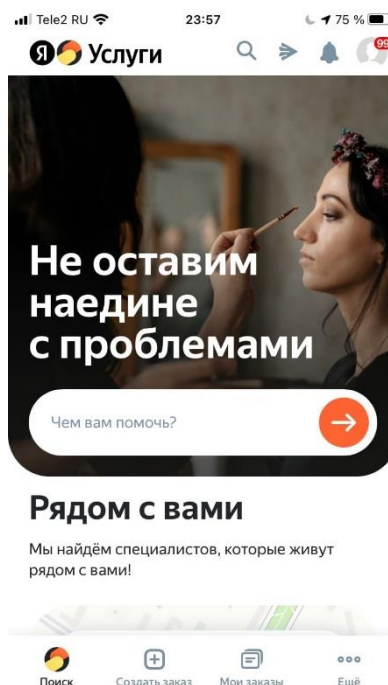


Рисунок 3 – Главный экран мобильного приложения «Яндекс. Услуги – уборка, ремонт»

Отличительной особенностью является то, что для размещения своей собственной услуги необходимо скачать приложение «Яндекс. Услуги для мастеров» и далее работать уже с ним [7]. Данная черта позволяет чётко разделить ответственность и помогает пользователю сосредоточиться на поиске исполнителя или заказа. На рисунке 4 представлен главный экран приложения «Яндекс. Услуги для мастеров».

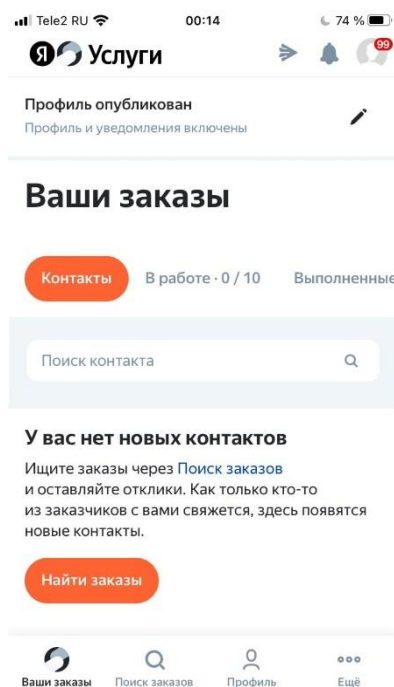


Рисунок 4 – Главный экран мобильного приложения «Яндекс. Услуги для мастеров»

Оба приложения отличаются высоким качеством исполнения, имеют приятный дизайн и дружелюбный пользовательский интерфейс.

К минусам приложения для заказчиков можно отнести отсутствие возможности просмотра исполнителей без публикации своего заказа.

Приложение для исполнителя имеет минус в виде сложного механизма идентификации при создании профиля, так как пользователю необходимо вводить много персональных данных, включая фотографию документа, подтверждающего личность. Приложение для исполнителя имеет довольно низку оценку в 3,6 баллов из 5, что свидетельствует о недовольстве пользователей. Также у обоих приложений довольно сложная система навигации.

Таким образом, было принято решение реализовать собственное мобильное приложение поддержки процессов взаимного поиска потребителей и поставщиков социальных услуг. Разрабатываемое приложение должно быть лишено выделенных недостатков аналогичных решений и учитывать их преимущества.

1.3 Анализ функциональных требований

Основная задача разрабатываемого мобильного приложения – это удобный поиск и размещение услуги.

На основе проведения анализа были выделены следующие основные функции мобильного приложения:

- войти в систему;
- зарегистрироваться;
- редактировать профиль;
- осуществить поиск услуги по типу;
- осуществить поиск услуги по городу и цене;
- просмотреть услугу;
- добавить услугу в избранное;
- поделиться услугой;
- написать исполнителю или позвонить;
- создать услугу;
- создать тендер на услугу;
- просмотреть свои праздники;
- просмотреть конкурсы к празднику;
- купить конкурсы к празднику.

На основе функциональных требований была разработана диаграмма основных вариантов использования мобильного приложения поддержки процессов взаимного поиска потребителей и поставщиков социальных услуг. Диаграмма представлена на рисунке 5.

Диаграмма отражает отношения между вариантами использования и актерами и описывает систему на концептуальном уровне [8]. Было выделено два типа актеров: пользователь и сервер, предоставляющий доступ к хранящимся на нем данным.

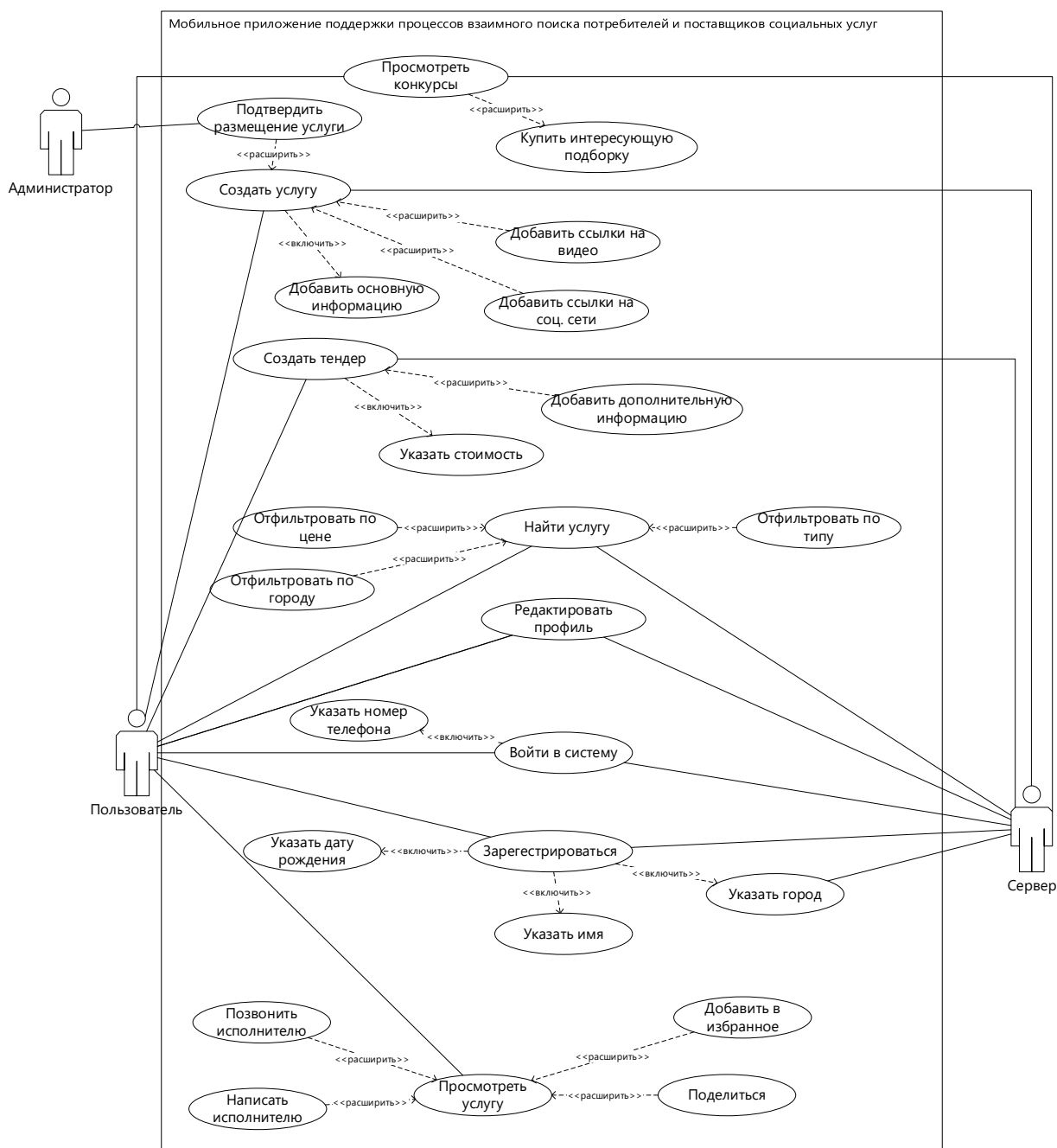


Рисунок 5 – Диаграмма вариантов использования мобильного приложения

Была разработана диаграмма деятельности для варианта использования «Редактировать профиль». Диаграмма деятельности позволяет подробно рассмотреть поведение вариантов использования. На ней представлены несколько дорожек и соответствующие им состояния. Рассмотреть диаграмму можно на рисунке 6.

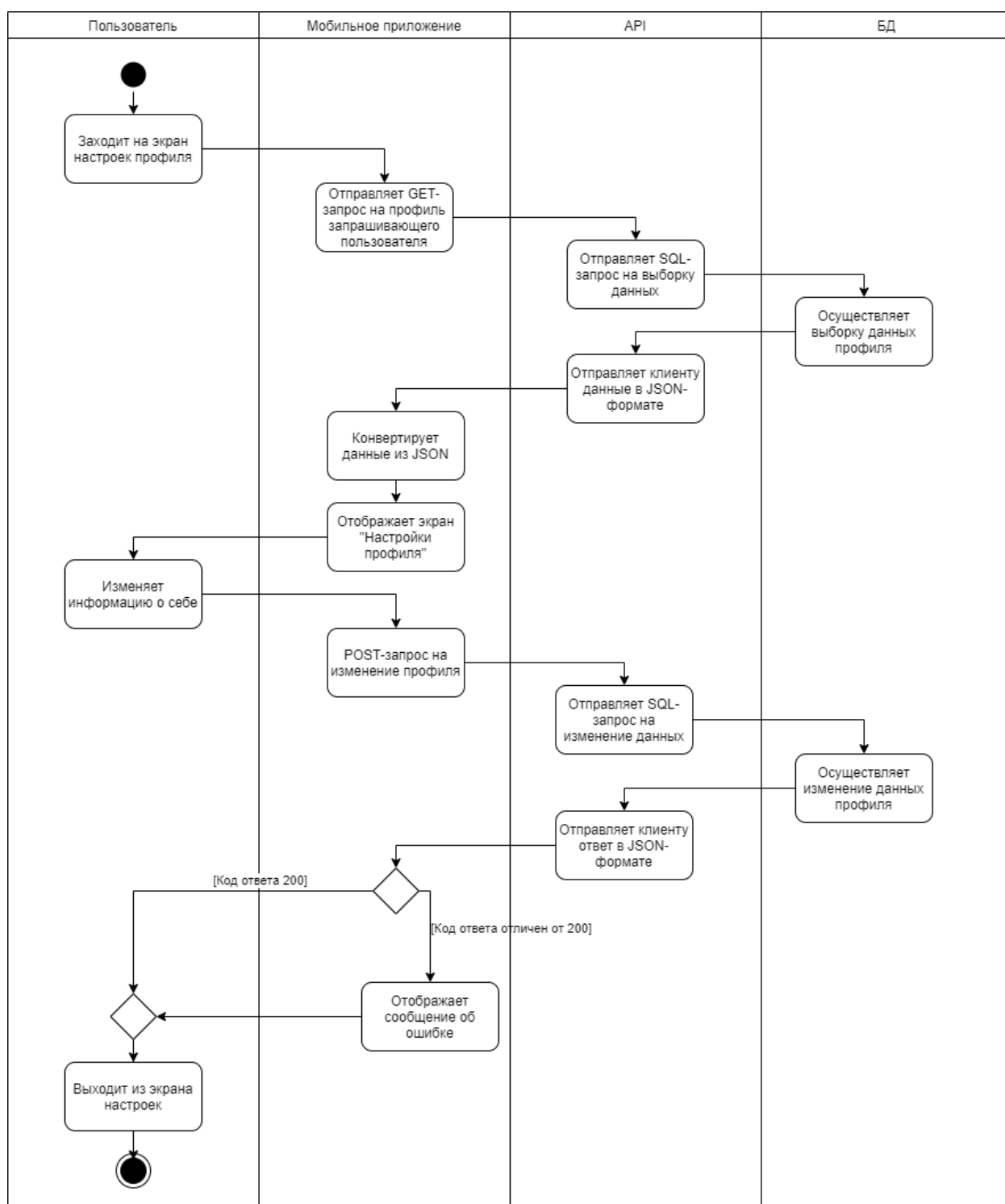


Рисунок 6 – Диаграмма деятельности для варианта использования «Редактировать профиль»

На рисунке 7 представлена диаграмма деятельности для варианта использования «Создать услугу».

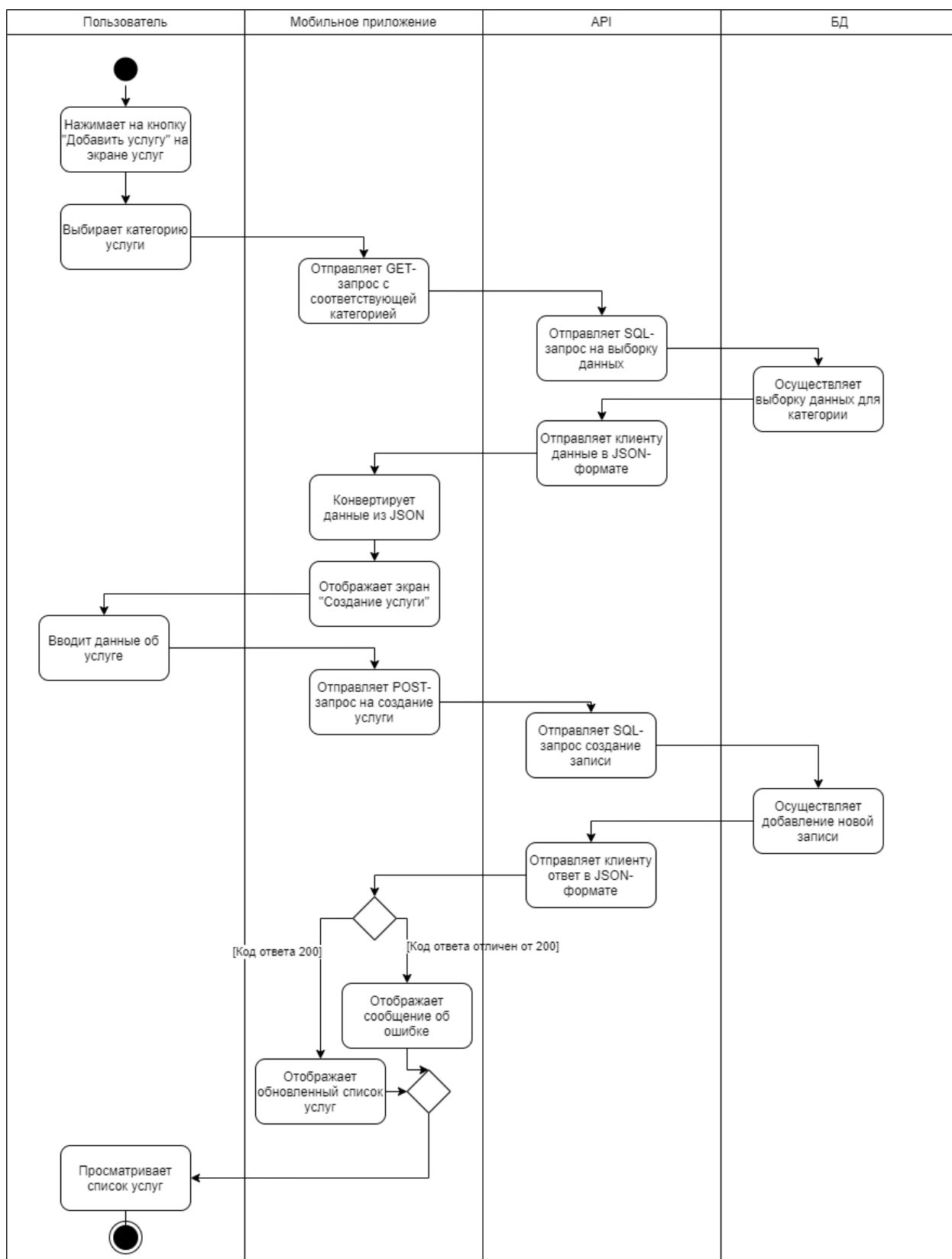


Рисунок 7 – Диаграмма деятельности для варианта использования «Создать услугу»

Для более детального рассмотрения логики вариантов использования были разработаны диаграммы последовательности. На рисунке 8 представлена диаграмма для варианта использования «Создать тендер».

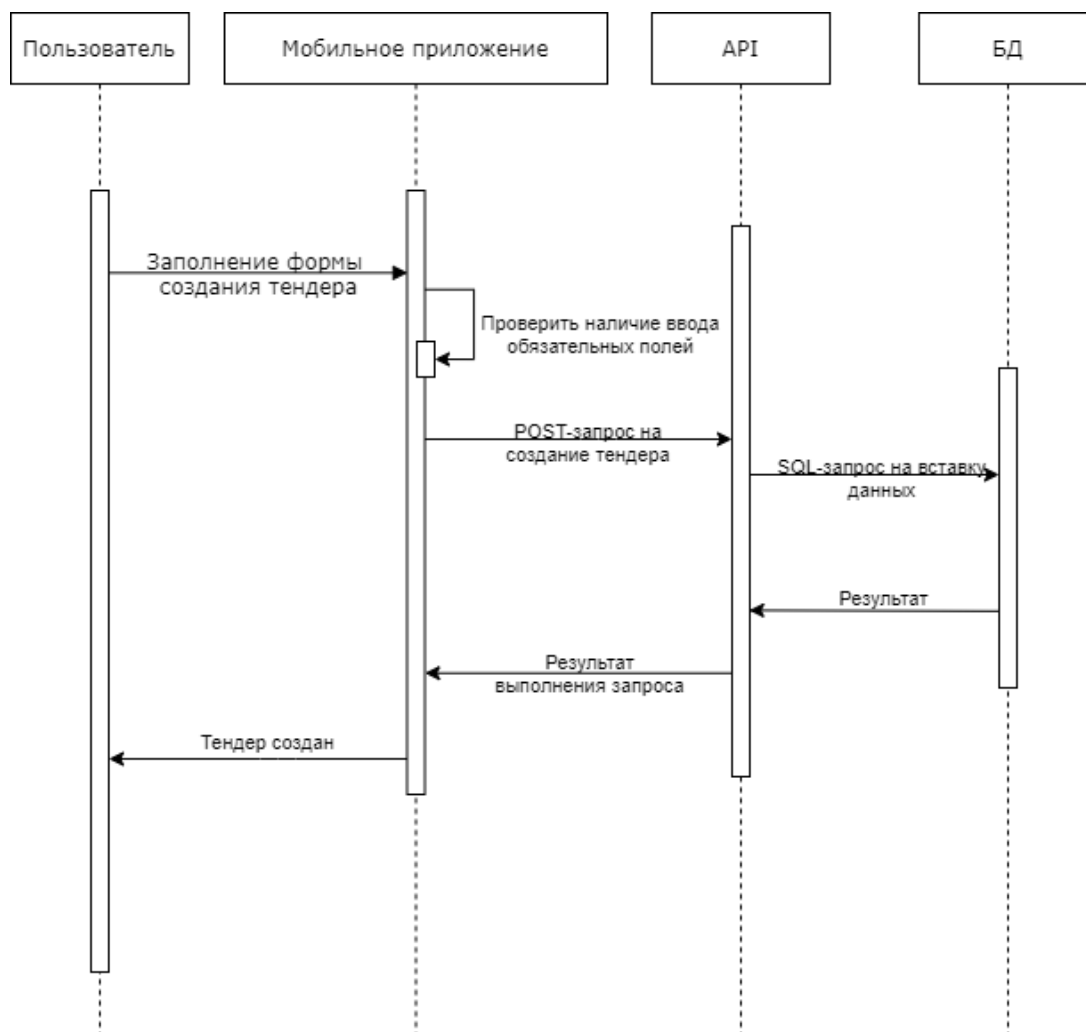


Рисунок 8 – Диаграмма последовательности для варианта использования «Создать тендер»

Диаграмма последовательности позволяет уточнить взаимодействие между актерами и элементами системы в рамках рассматриваемого варианта использования.

1.4 Анализ нефункциональных требований

Нефункциональные требования необходимо зафиксировать как для клиентской части, так и для серверной.

Требования к клиентской части мобильного приложения:

- приложение должно быть удобным в использовании и обладать интуитивно понятным интерфейсом;
- приложение должно быть отказоустойчивым;
- приложение должно быть расширяемым в случае появления новых функциональных требований;
- приложение должно быть безопасным для пользователя;
- совместимость с операционной системой iOS 9.

Требования к серверной части:

- серверная часть должна представлять из себя REST Application Programming Interface (API, интерфейс прикладного программирования);
- формат данных для обмена информацией между клиентской частью и сервером – JSON;
- СУБД MySql Server.

1.5 Построение исходной концептуальной модели предметной области

На основе анализа предметной области и функциональных требований была построена исходная концептуальная модель. Выделены типы сущностей, а также их свойства и типы связей между ними.

Описание типов сущностей представлено в таблице 1.

Таблица 1 – Типы сущностей предметной области

№	Наименование	Описание	Категория
1	Пользователь	Список пользователей	Стержневая
2	Тендер	Список тендеров	Стержневая
3	Изображение	Список изображений	Стержневая
4	Конкурс	Список конкурсов	Стержневая
5	Подборка конкурсов	Список конкурсных подборок	Стержневая
6	Отзыв на подборку конкурсов	Список отзывов к конкурсным подборкам	Стержневая
7	Город	Список городов	Стержневая
8	Токен авторизации	Список авторизационных токенов	Стержневая
9	Тип услуги	Список типов услуг	Стержневая
10	Видео	Список ссылок на видеоленты	Стержневая

Продолжение таблицы 1

№	Наименование	Описание	Категория
11	Социальная сеть	Список социальных сетей и ссылок на них	Стержневая
12	Отзыв на услугу	Список отзывов на услугу	Стержневая
13	Фото услуги	Список фотографий к услуге	Стержневая
14	Отметка «Мне нравится»	Список отметок «Мне нравится» к услуге	Стержневая
15	Избранное	Список избранных услуг	Стержневая
16	Услуга	Список услуг	Стержневая

В таблице 2 представлено описание типов связей между типами сущностей предметной области.

Таблица 2 – Типы связей между типами сущностей

№	Тип сущности 1	Тип сущности 2	Описание	Вид связи	Обязательность
1	Тип услуги	Тендер	Тендеру соответствует определенный тип услуги	1 : М	Обязательно
2	Тип услуги	Услуга	У услуги есть определенный тип	1 : М	Обязательно
3	Город	Пользователь	Пользователю соответствует город	1 : М	Обязательно
4	Город	Услуга	Услуге соответствует город	1 : М	Обязательно
5	Услуга	Видео	Услуге соответствует видео	1 : М	Необязательно
6	Услуга	Социальная сеть	Услуге соответствует социальная сеть	1 : М	Необязательно
7	Подборка конкурсов	Конкурс	Подборке конкурсов соответствует конкурс	1 : М	Обязательно
8	Подборка конкурсов	Отзыв на подборку конкурсов	Подборка конкурсов может иметь отзыв	1 : М	Необязательно
9	Изображение	Фото услуги	Изображение соответствует фотографии услуги	1 : 1	Обязательно

Продолжение таблицы 2

№	Тип сущности 1	Тип сущности 2	Описание	Вид связи	Обязательность
10	Изображение	Подборка конкурсов	Изображение соответствует подборке конкурсов	1 : 1	Необязательно
11	Пользователь	Услуга	Пользователю соответствует услуга	1 : M	Необязательно
12	Пользователь	Тендер	Пользователю соответствует тендер	1 : M	Необязательно
13	Пользователь	Отзыв на подборку конкурсов	Пользователю соответствует отзыв на подборку конкурсов	1 : M	Необязательно
14	Пользователь	Токен авторизации	Пользователю соответствует токен авторизации	1 : 1	Необязательно
15	Пользователь	Изображение	Пользователю соответствует изображение	1 : 1	Необязательно
16	Пользователь	Избранное	Пользователю соответствует избранная услуга	1 : M	Необязательно
17	Пользователь	Отзыв на услугу	Пользователю соответствует отзыв на услугу	1 : M	Необязательно
18	Пользователь	Отметка «Мне нравится»	Пользователю соответствует отметка «Мне нравится»	1 : M	Необязательно
19	Услуга	Избранное	Услуга соответствует избранному	1 : M	Необязательно
20	Услуга	Отзыв на услугу	Услуга соответствует отзыву на услугу	1 : M	Необязательно
21	Услуга	Отметка «Мне нравится»	Услуга соответствует отметке «Мне нравится»	1 : M	Необязательно
22	Услуга	Фото услуги	Услуга соответствует фотографии услуги	1 : M	Обязательно

Концептуальная модель предметной области представлена в нотации IDEF1 на рисунке 9.

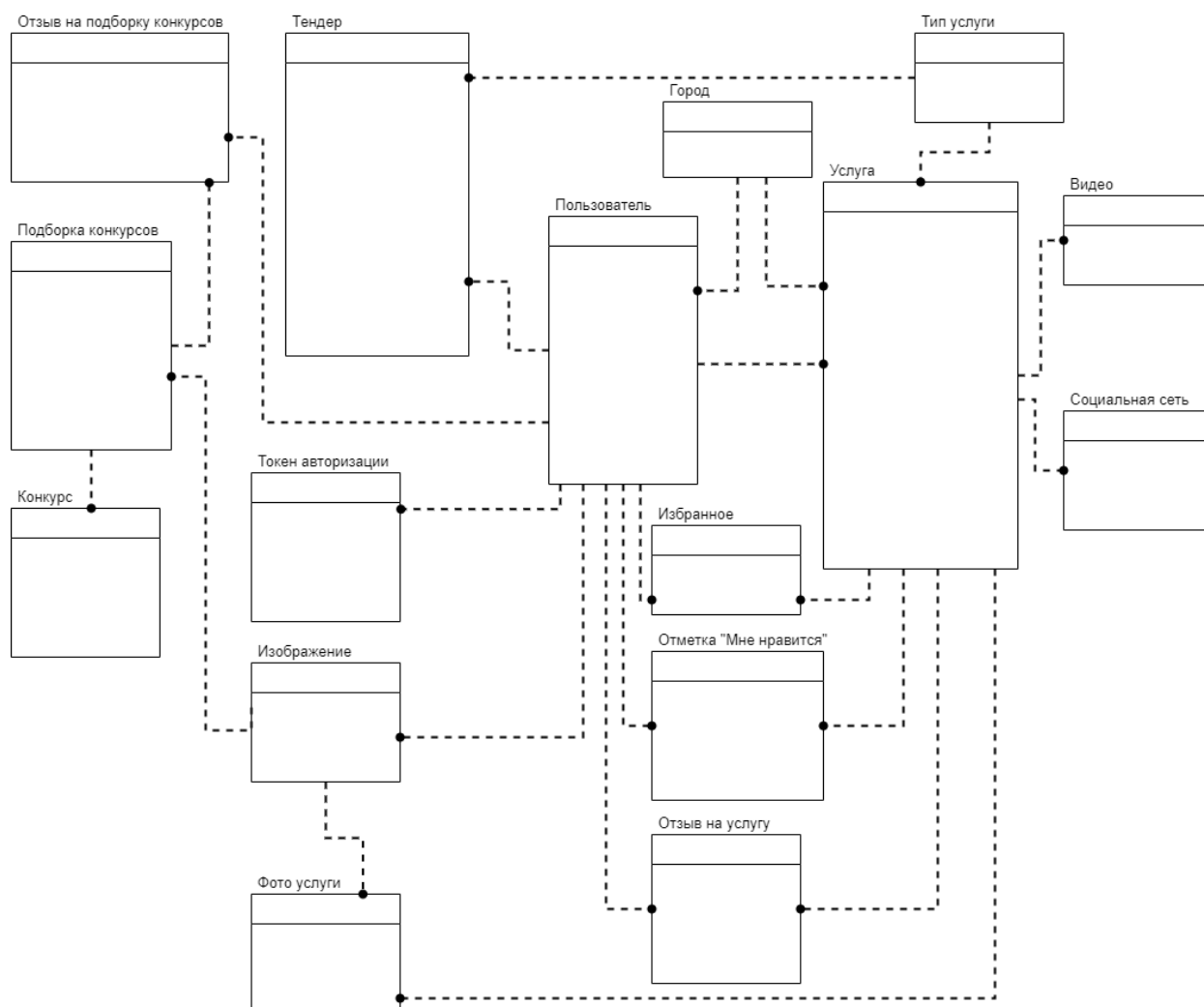


Рисунок 9 – Исходная концептуальная модель предметной области

Стандарт IDEF1 является аналитическим методом и позволяет построить наглядную модель информационной структуры приложения [9].

2 Проектирование мобильного приложения

2.1 Выбор технологий и инструментальных средств разработки

Разработка будет производиться по технологии Rational Unified Process (RUP). RUP предполагает итерационный подход к разработке программного обеспечения.

Проект делится на совокупность небольших проектов, выполняющихся друг за другом. Для каждой итерации определяются цели, которые необходимо достичь, и по окончании должна быть получена промежуточная версия конечного продукта. Таким образом, данный подход позволяет обнаружить ошибки на ранних этапах разработки, что способствует снижению коммерческих рисков [10].

В процессе объектно-ориентированного анализа и проектирования использовался универсальный язык моделирования UML. С помощью синтаксиса UML можно разрабатывать различные диаграммы, моделирующие бизнес-процессы, структуры систем.

Разработка мобильного приложения будет осуществляться для операционной системы iOS, соответственно, принято решение использовать технологии, предоставленные компанией Apple. Разработка будет осуществляться на языке программирования Swift 5.0 в среде Xcode 12.0.

Тестирование мобильного приложения производилось на смартфоне iPhone SE с версией операционной системы iOS 14.4.2.

2.2 Планирование разработки мобильного приложения и оценка бюджета

В ходе планирования мобильного приложения поддержки процессов взаимного поиска потребителей и поставщиков социальных услуг было выделено 33 работы. Они представлены в таблице 3. Для каждой задачи выделены вид, длительность и предшествующие работы.

Таблица 3 – Перечень работ проекта

№	Название		Длительность	Предшественники
1	Начало проекта	Веха		-
2	Анализ предметной области и требований	Фаза		-
3	Анализ приложений-аналогов	Задача	14	1
4	Анализ рынка социальных услуг	Задача	7	3
5	Консультация с поставщиками и потребителями социальных услуг	Задача	7	4
6	Принятие окончательного решения	Задача	2	5
7	Анализ требований завершен	Задача	-	6
8	Выбор средств реализации проекта	Фаза	-	-
9	Анализ возможных программных средств	Задача	4	1
10	Выбор программного решения для реализации приложения	Задача	4	9
11	Выбор программного решения завершен	Задача	-	10
12	Проектирование приложения	Фаза	-	-
13	Проектирование структур данных	Задача	4	7
14	Проектирование архитектуры приложения	Задача	7	7;11
15	Проектирование пользовательского интерфейса	Задача	7	7
16	Реализация приложения	Фаза	-	-
17	Реализация структур данных	Задача	4	13
18	Реализация запросов к серверу	Задача	7	14;17
19	Реализация пользовательского интерфейса	Задача	16	15;18
20	Реализация проекта завершена	Задача	-	19
21	Тестирование приложения	Фаза	-	-
22	Тестирование запросов к серверу	Задача	4	18
23	Тестирование пользовательского интерфейса	Задача	4	19
24	Исправление возникших ошибок	Задача	7	22; 23
25	Тестирование завершено	Задача	-	24
26	Выпуск приложения	Фаза	-	-
27	Поиск рекламных площадок	Задача	4	25
28	Поиск заинтересованных поставщиков социальных услуг	Задача	4	25

Продолжение таблицы 3

№	Название	Вид задачи	Длительность	Предшественники
29	Выкладка приложения	Задача	3	27;28
30	Анализ обратной связи пользователей	Задача	7	29
31	Исправление обнаруженных недочетов	Задача	7	30
32	Выпуск приложения завершен	Задача	-	31
33	Конец проекта	Веха	-	32

В Microsoft Office Project был создан проект с датой начала 06.06.2021 и были занесены все выделенные задачи [11].

На следующем этапе требуется определить ресурсы, необходимые для выполнения каждой задачи проекта, и затраты на них (таблица 4).

Таблица 4 – Ресурсы проекта

Название ресурса	Тип	Макс. единиц	Затраты		Затраты на исп.
			Станд. Ставка	Ставка сверхур.	
Разработчик мобильных приложений	Трудовой	300%	250 р./ч	300 р./ч	-
Управляющий проектом	Трудовой	100%	200 р./ч	250 р./ч	-
Дизайнер	Трудовой	100%	150 р./ч	200 р./ч	-
Аренда офиса	Затраты				-
Интернет	Затраты				-
Ноутбук	Материальный		100000 р.		-

Из представленной таблицы видно, что реализацией проекта будут заниматься трое разработчиков мобильных приложений, управляющий проектом и дизайнер, которым для работы нужны ноутбуки, доступ к интернету и арендованный офис.

Занесём все ресурсы в Microsoft Office Project. Результаты представлены на рисунке 10.

	Название ресурса	Тип	Единицы измерения материала	Краткое название	Макс. единиц	Стандартная ставка	Ставка сверхурочн	Затраты на использ.	Начисление	Базовый календарь
1	Управляющий проектом	Трудовой		У	100%	200,00 Р/ч	0,00 Р/ч	0,00 Р	Пропорциональное	Стандартный
2	Разработчик мобильных приложений	Трудовой		Р	300%	250,00 Р/ч	0,00 Р/ч	0,00 Р	Пропорциональное	Стандартный
3	Дизайнер	Трудовой		Д	100%	150,00 Р/ч	0,00 Р/ч	0,00 Р	Пропорциональное	Стандартный
4	Аренда офиса	Затраты		А					Пропорциональное	
5	Интернет	Затраты		И					Пропорциональное	
6	Ноутбук	Материальный	Единиц	Н		100 000,00 Р		0,00 Р	Пропорциональное	

Рисунок 10 – Лист ресурсов

Закрепим ресурсы за задачами, и в результате получим иллюстрацию календарного плана в виде диаграммы Ганта, представленную на рисунке 11.

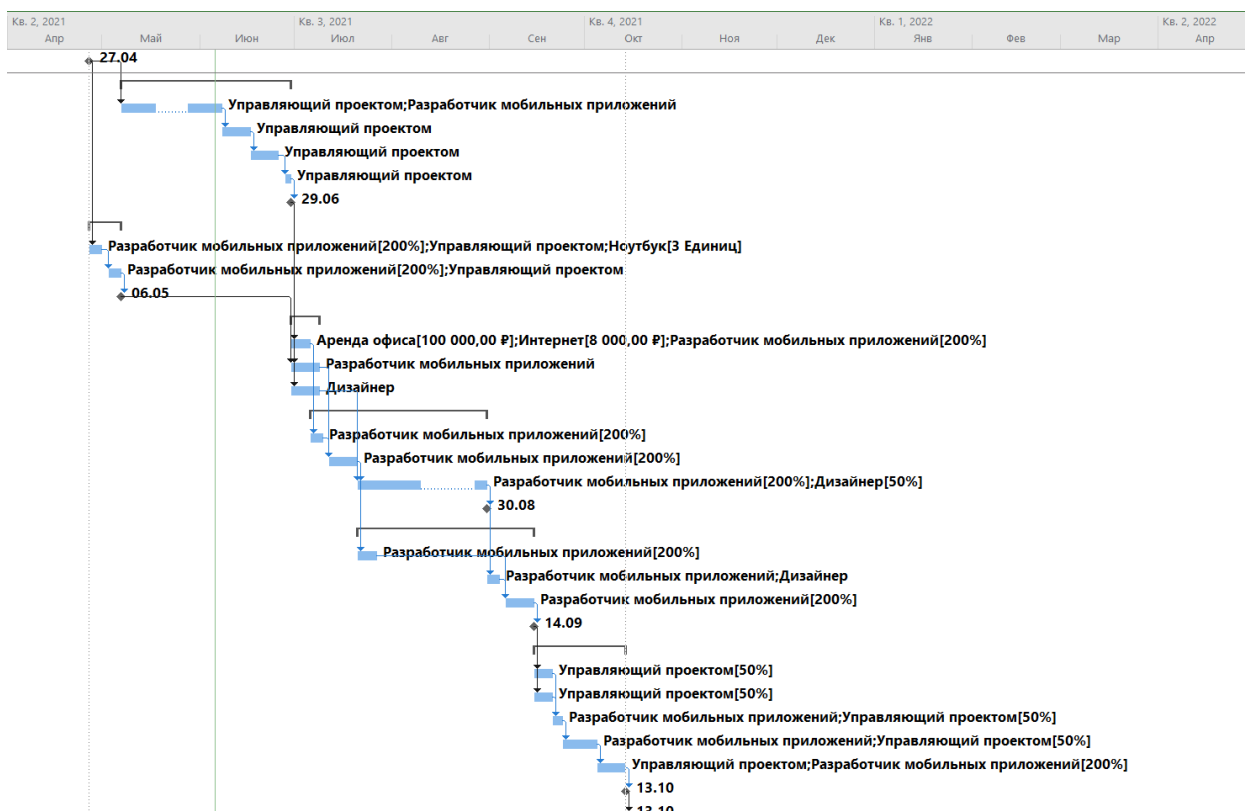


Рисунок 11 – Диаграмма Ганта

Далее был получен обзор проекта разработки мобильного приложения поддержки процессов взаимного поиска потребителей и поставщиков социальных услуг, представленный на рисунке 12. Из рисунка видно, что длительность выполнения всего проекта составляет 122 дня, а затраты на реализацию – 771600 рублей.

Мобильное приложение поддержки процессов взаимного поиска потребителей и поставщиков социальных услуг

на дату Вс 06.06.21

Даты			
Начало:	Вт 27.04.21	Окончание:	Ср 13.10.21
Баз. начало:	НД	Баз. окончание:	НД
Факт. начало:	НД	Факт. окончание:	НД
Отклон. начала:	0 дней	Отклон. окончания:	0 дней
Длительность			
Текущий план:	122 дней	Оставшаяся:	122 дней
Базовый план:	0 дней?	Фактическая:	0 дней
Отклонение:	122 дней	Процент завершени	0%
Трудозатраты			
Текущий план:	1 584 ч	Оставшиеся:	1 584 ч
Базовый план:	0 ч	Фактические:	0 ч
Отклонение:	1 584 ч	Процент завершени	0%
Затраты			
Текущий план:	771 600,00 Р	Оставшиеся:	771 600,00 Р
Базовый план:	0,00 Р	Фактические:	0,00 Р
Отклонение:	771 600,00 Р		
Состояние задачи		Состояние ресурса	
Еще не начатые задачи:	33	Трудовые ресурсы:	5
Выполняющиеся задачи	0	Трудовые ресурсы с прев. дост	0
Завершенные задачи:	0	Материальные ресурсы:	1
Итого задач:	33	Итого ресурсов:	6

Рисунок 12 – Обзор проекта

Можно сделать вывод, что разработка мобильного приложения поддержки процессов взаимного поиска потребителей и поставщиков социальных услуг является экономически целесообразной.

2.3 Проектирование пользовательского интерфейса

На основе функциональных требований спроектирован графический интерфейс пользователя мобильного приложения поддержки процессов взаимного поиска потребителей и поставщиков социальных услуг. При

проектировании соблюдались рекомендации для мобильных приложений платформы iOS, данные компанией Apple в руководстве Human Interface Guidelines [12].

Графический интерфейс пользователя включает в себя следующие основные экраны:

- «Главный экран»;
- экран «Вход»;
- экран «Подтверждение номера»;
- экран «Регистрация»;
- экран «Боковое меню»;
- экран «Профиль пользователя»;
- экран «Настройки профиля»;
- экран «Услуги пользователя»;
- экран «Создание услуги»;
- экран «Каталог исполнителей»;
- экран «Исполнители конкретной услуги»;
- экран «Фильтры услуг»;
- экран «Услуга»;
- экран «Тендеры»;
- экран «Создание тендера».

На рисунке 13 изображен главный экран мобильного приложения. С помощью него можно связаться с технической поддержкой по нажатию на кнопку в центре, а также осуществить переход на другие экраны, такие как «Тендеры» и «Каталог исполнителей». Также на экране доступна смена текущего города и поиск услуг по ключевым словам. Кнопка связи с технической поддержкой имеет анимацию в виде повторяющегося градиентного переливания.

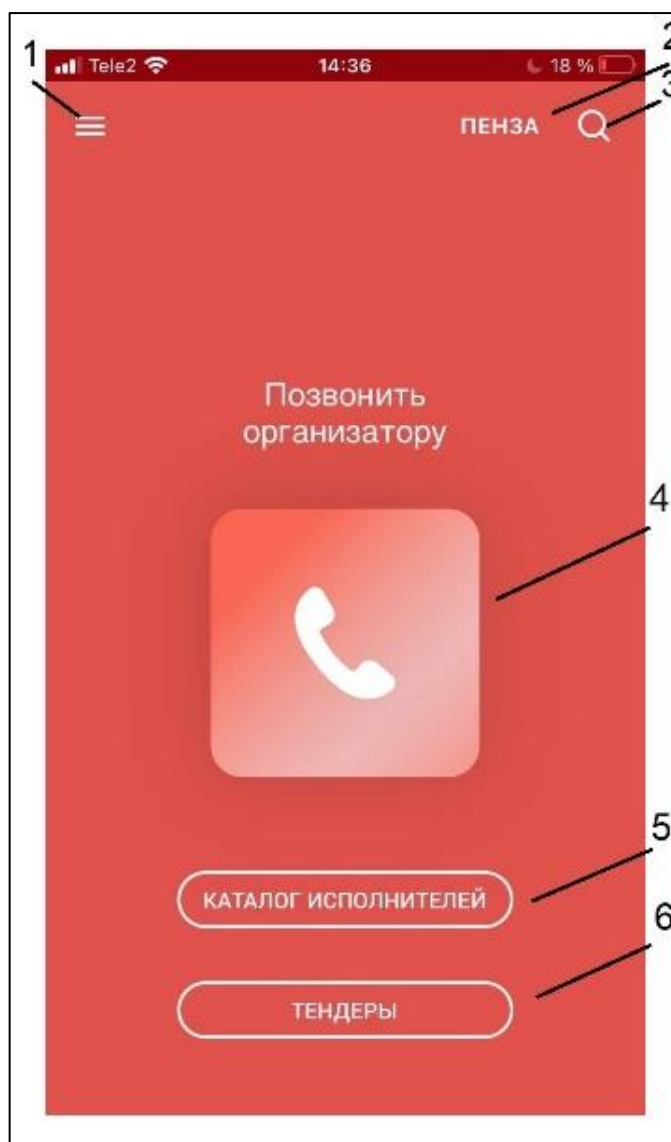


Рисунок 13 – «Главный экран»

Рассмотрим основные элементы интерфейса данного экрана:

- 1 – кнопка вызова бокового меню;
- 2 – кнопка выбора города;
- 3 – кнопка поиска исполнителей по ключевым словам;
- 4 – анимированная кнопка телефонного вызова технической поддержки;
- 5 – кнопка перехода к каталогу исполнителей;
- 6 – кнопка перехода к тендерам.

На рисунке 14 представлен экран «Вход» для авторизации пользователя.

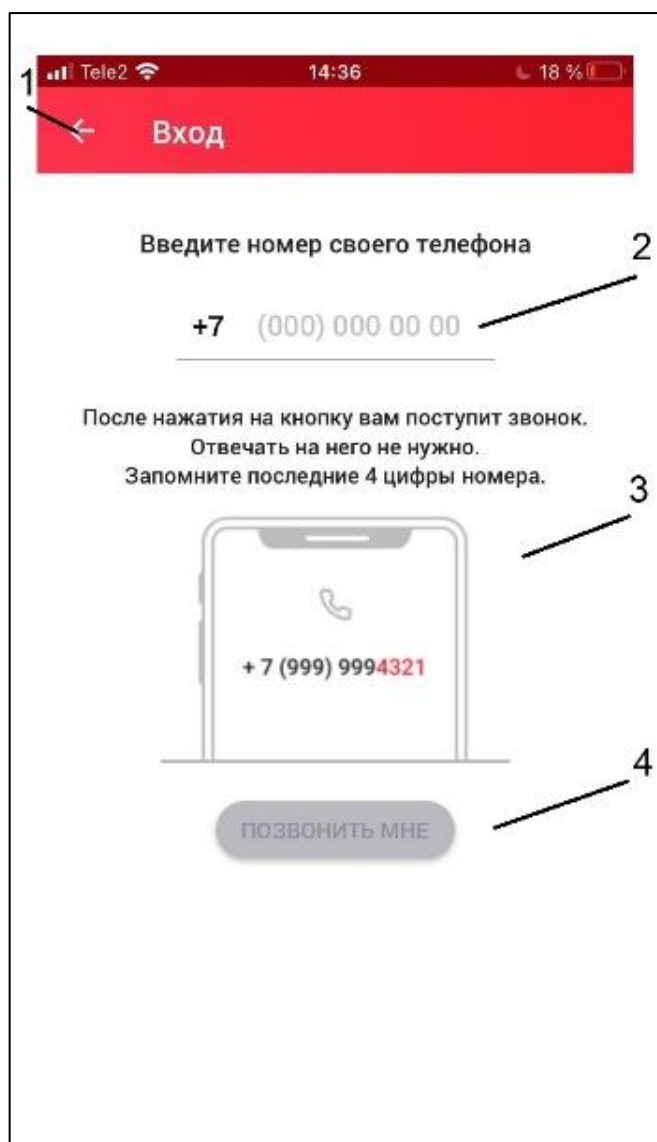


Рисунок 14 – Экран «Вход»

Основные элементы экрана «Вход»:

- 1 – кнопка возврата на главный экран;
- 2 – поле ввода номера телефона пользователя;
- 3 – текстовое поле с картинкой, выступающее в роли инструкции для пользователя;
- 4 – кнопка перехода для дальнейшей авторизации.

На рисунке 15 изображен интерфейс экрана «Подтверждение номера». Пользователь попадает на него в случае, если введенный ранее номер телефона есть в базе пользователей приложения.

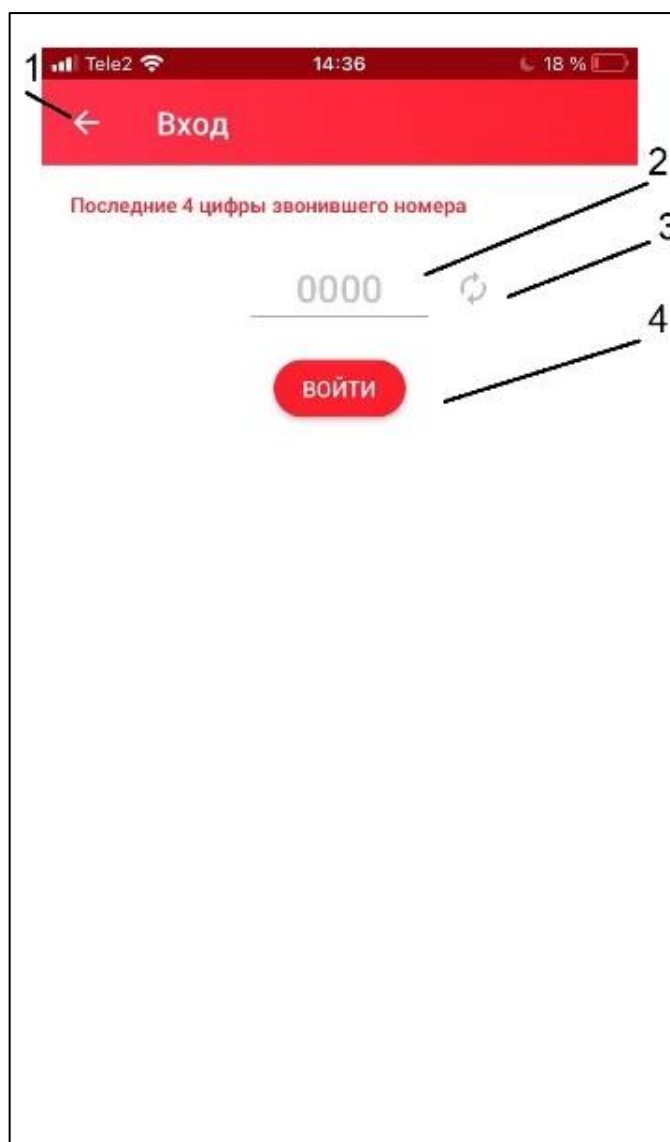


Рисунок 15 – Экран «Подтверждение номера»

Рассмотрим основные элементы интерфейса данного экрана:

- 1 – кнопка возврата на экран «Вход»;
- 2 – поле ввода цифр для подтверждения номера телефона;
- 3 – кнопка повторного подтверждения номера телефона;
- 4 – кнопка перехода на главный экран.

На рисунке 16 представлен экран «Регистрация». Пользователь попадает на него в случае отсутствия профиля для введённого ранее мобильного телефона.

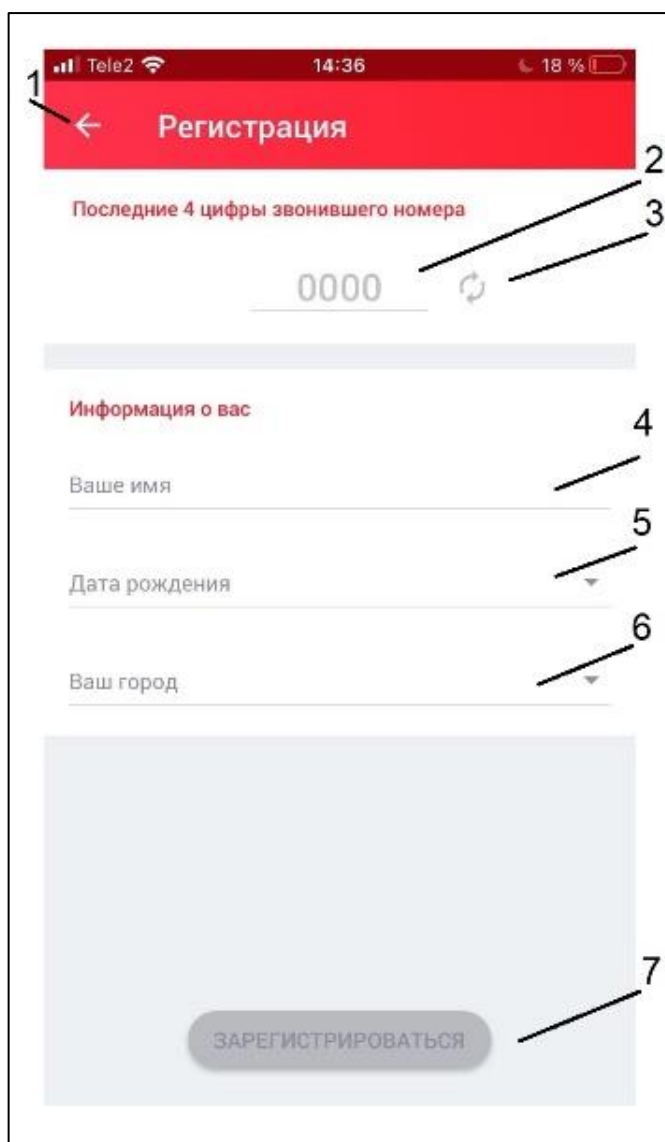


Рисунок 16 – Экран «Регистрация»

На рисунке 16 отмечено:

- 1 – кнопка возврата на экран «Вход»;
- 2 – поле ввода цифр для подтверждения номера телефона;
- 3 – кнопка повторного подтверждения номера телефона;
- 4 – поле ввода имени пользователя;
- 5 – поле ввода даты рождения пользователя;
- 6 – поле ввода города пользователя;
- 7 – кнопка подтверждения регистрации пользователя.

На рисунке 17 изображен экран «Боковое меню».

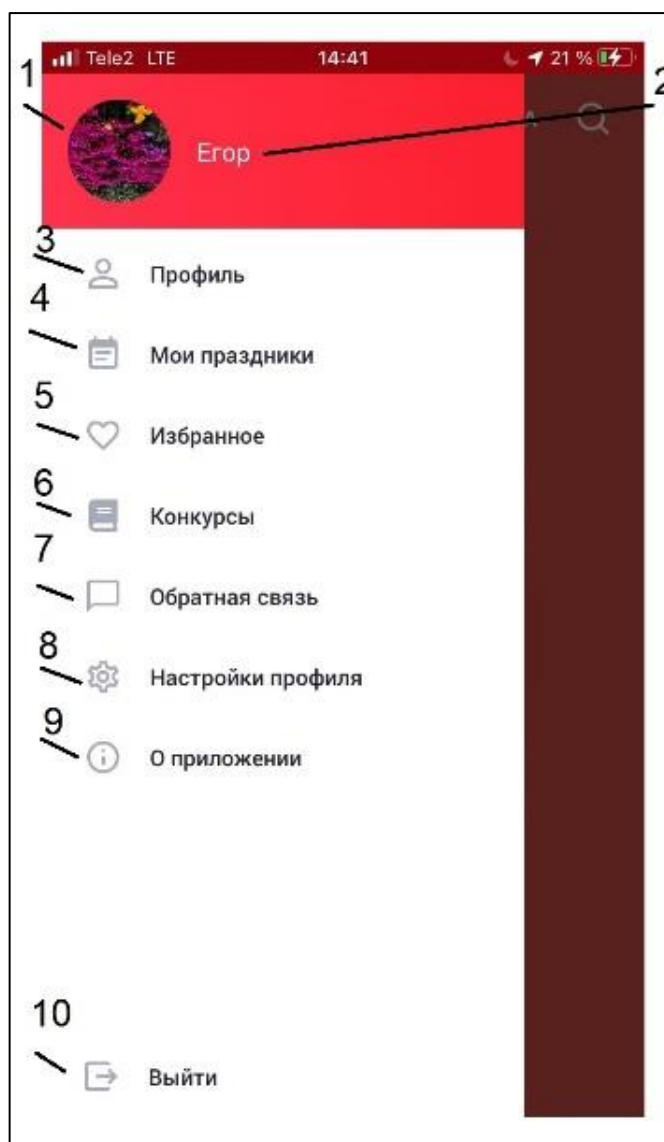


Рисунок 17 – Экран «Боковое меню»

На рисунке 17 отмечено:

- 1 – текущий аватар пользователя;
- 2 – текстовое поле с именем пользователя;
- 3 – кнопка перехода к профилю пользователя;
- 4 – кнопка перехода к праздникам пользователя;
- 5 – кнопка перехода к избранным услугам;
- 6 – кнопка перехода к списку подборок конкурсов;
- 7 – кнопка перехода на экран обратной связи;
- 8 – кнопка перехода к настройкам профиля пользователя;
- 9 – кнопка перехода на экран информации о приложении;

10 – кнопка выхода из текущего профиля.

На рисунке 18 изображен экран «Профиль пользователя».

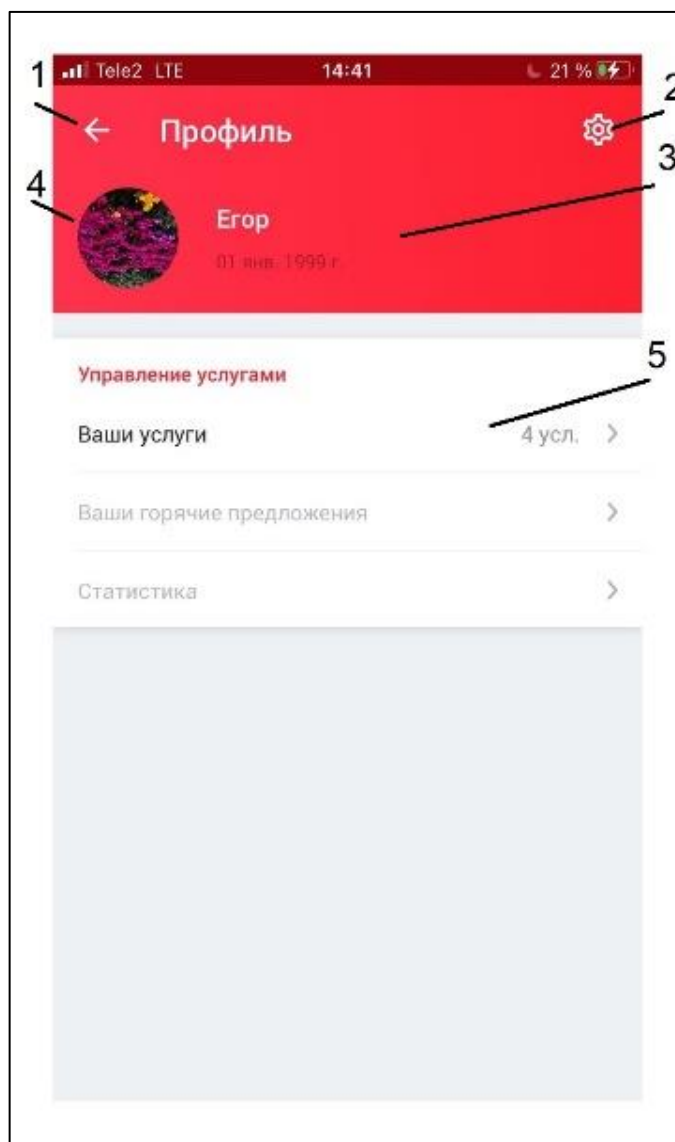


Рисунок 18 – Экран «Профиль пользователя»

Рассмотрим основные элементы данного экрана:

- 1 – кнопка возврата к предыдущему экрану;
- 2 – кнопка перехода на экран «Настройки профиля»;
- 3 – текстовые поля с текущим именем и датой рождения пользователя;
- 4 – текущий аватар пользователя;
- 5 – кнопка перехода на экран «Услуги пользователя».

На рисунке 19 изображен экран «Настройки профиля». На него можно попасть из бокового меню или из экрана «Профиль пользователя». Поля для ввода информации о пользователе заранее заполнены текущими данными.



Рисунок 19 – Экран «Настройки профиля»

На рисунке 19 отмечено:

- 1 – кнопка возврата к предыдущему экрану;
- 2 – кнопка выбора нового аватара;
- 3 – текущий аватар пользователя;
- 4 – поле ввода имени пользователя;
- 5 – поле ввода даты рождения пользователя.

На рисунке 20 изображен интерфейс экрана «Услуги пользователя».

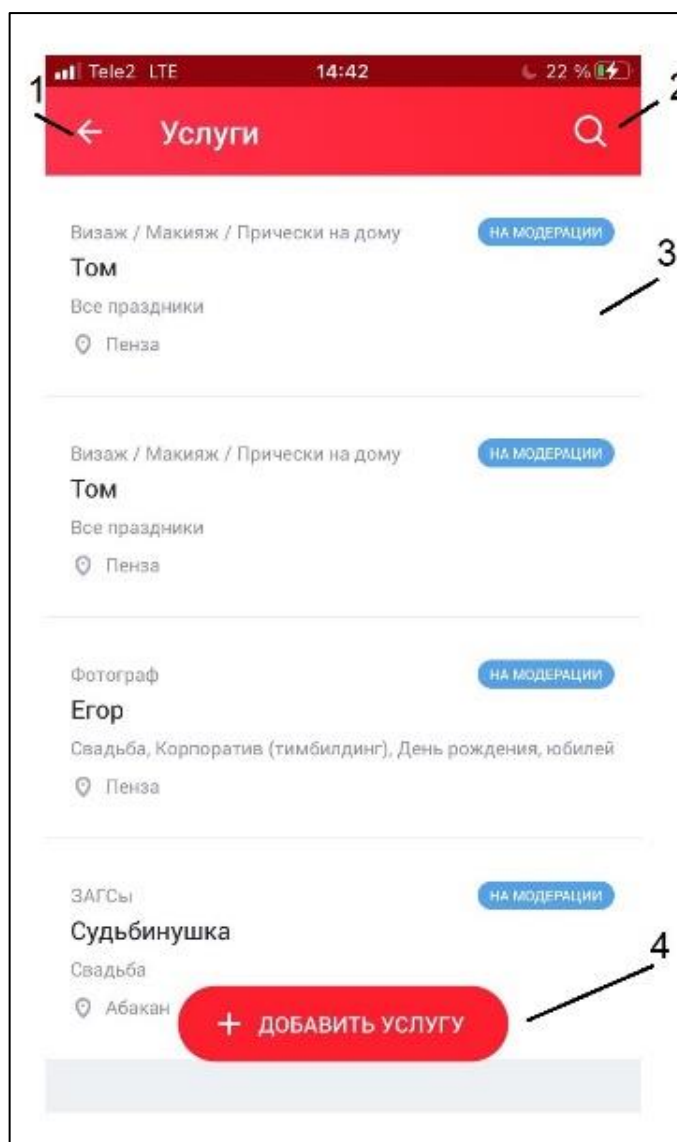


Рисунок 20 – Экран «Услуги пользователя»

На рисунке 20:

- 1 – кнопка возврата на предыдущий экран;
- 2 – кнопка вызова поля ввода для поиска услуги пользователя по названию;
- 3 – список услуг пользователя;
- 4 – кнопка перехода на экран «Создание услуги».

Экран «Создание услуги» представлен на рисунке 21. Данный экран имеет обязательный блок с основной информацией об услуге, а также динамический, содержание которого зависит от категории выбранной пользователем услуги.

1

← Создание услуги

Категория услуги

Выберите категорию услуги

Ведущий / Тамада

Общая информация

Отображаемое имя для услуги

Егор

Контактный телефон

+7 (906) 395-97-25

Город оказания услуги

Пенза

Адрес оказания услуги (необязательно)

Описание услуги

Аватар для этой услуги

Рисунок 21 – Экран «Создание услуги»

Рассмотрим основные компоненты интерфейса экрана «Создание услуги»:

- 1 – кнопка возврата к предыдущему экрану;
- 2 – кнопка выбора категории услуги;
- 3 – поле ввода названия услуги;
- 4 – поле ввода мобильного телефона создателя услуги;
- 5 – кнопка выбора города оказания услуги;
- 6 – поле ввода конкретного адреса оказания услуги;
- 7 – поле ввода описания услуги.

На рисунке 22 представлено продолжение экрана «Создание услуги».

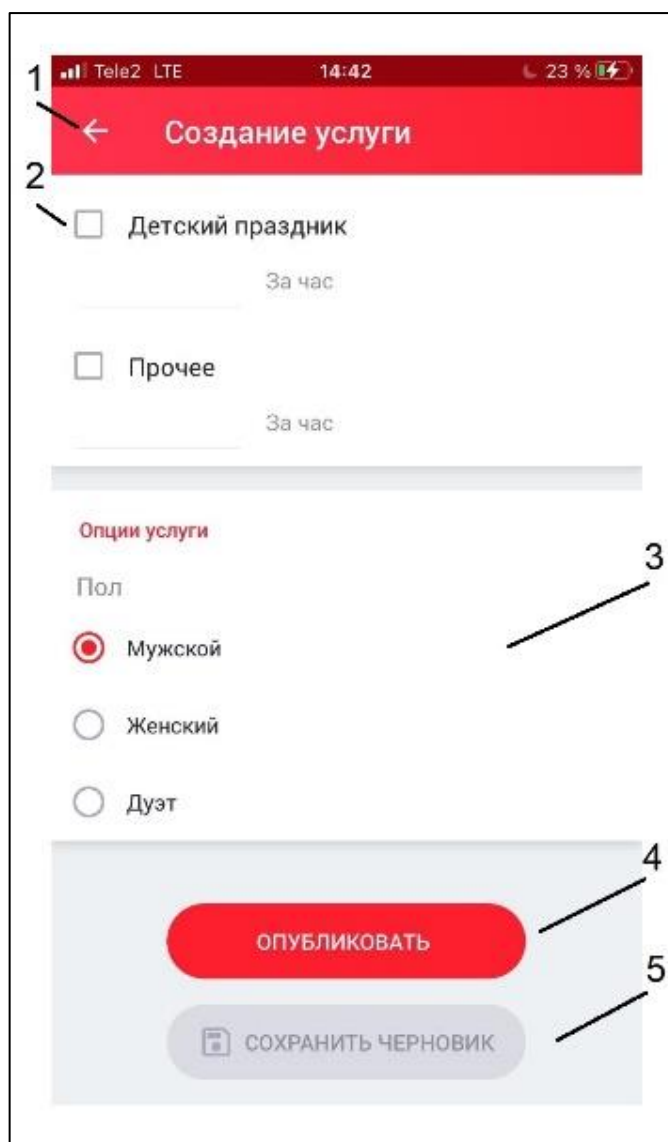


Рисунок 22 – Продолжение экрана «Создание услуги»

На рисунке 22:

- 1 – кнопка возврата на предыдущий экран;
- 2 – поля выбора типа мероприятия и их стоимости;
- 3 – поля выбора пола исполнителя услуги;
- 4 – кнопка отправки услуги на модерацию;
- 5 – кнопка сохранения услуги в черновики.

На рисунке 23 изображен экран «Каталог исполнителей». На него можно попасть с главного экрана приложения.



Рисунок 23 – Экран «Каталог исполнителей»

Основные компоненты данной страницы:

- 1 – кнопка возврата на предыдущий экран;
- 2 – кнопка выбора типа праздника;
- 3 – полный список категорий услуг.

Интерфейс экрана «Исполнители конкретной услуги» представлен на рисунке 24. Наименование категории услуги на этом экране меняется в зависимости от её выбора на экране «Каталог исполнителей».

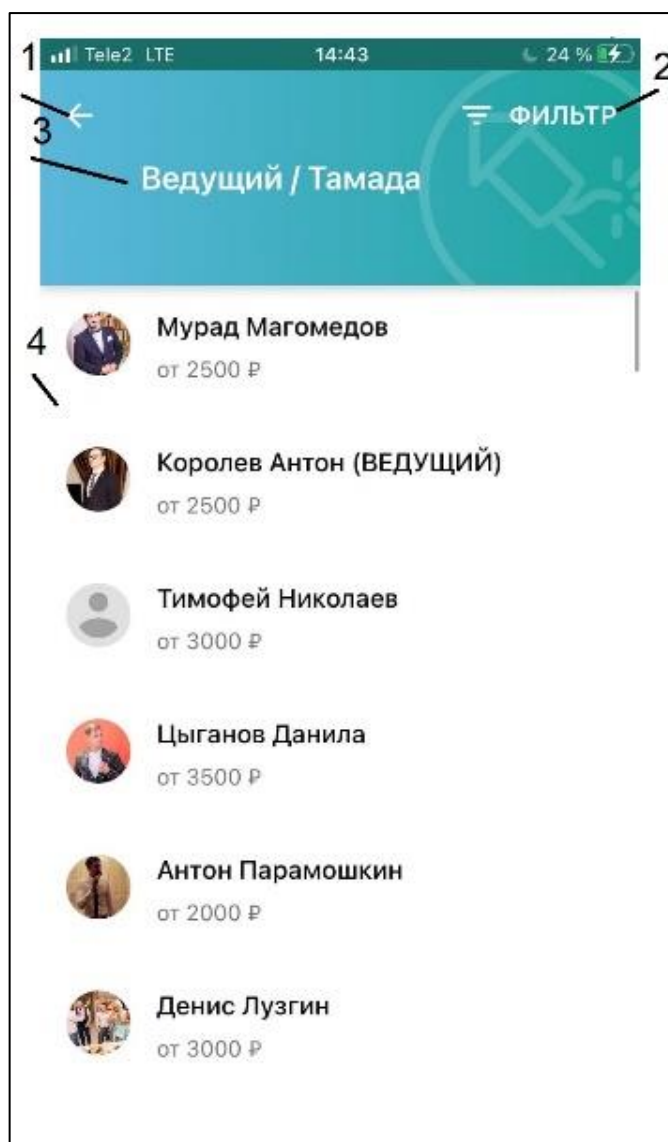


Рисунок 24 – Экран «Исполнители конкретной услуги»

Рассмотрим основные компоненты интерфейса экрана «Исполнители конкретной услуги»:

- 1 – кнопка возврата на предыдущий экран;
- 2 – кнопка перехода к экрану «Фильтры услуг»;
- 3 – текстовое поле, отображающее наименование категории услуги;
- 4 – список услуг.

На рисунке 25 изображен экран «Фильтры услуг». На кнопке подтверждения применения фильтров отображается количество найденных исполнителей.

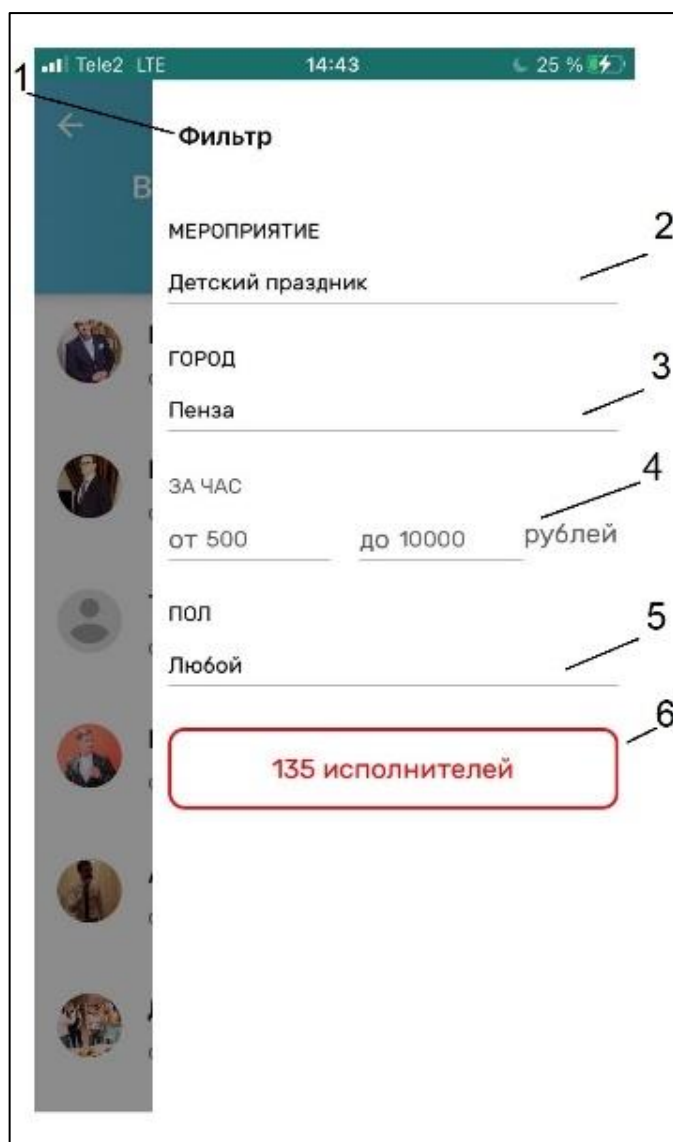


Рисунок 25 – Экран «Фильтры услуг»

На рисунке 25:

- 1 – текстовое поле наименования экрана;
- 2 – кнопка выбора типа мероприятия;
- 3 – кнопка выбора города оказания услуги;
- 4 – поля ввода диапазона цен для услуги;
- 5 – кнопка выбора пола исполнителя услуги;
- 6 – кнопка показа услуг после применения фильтров.

Экран «Услуга» изображен на рисунке 26.

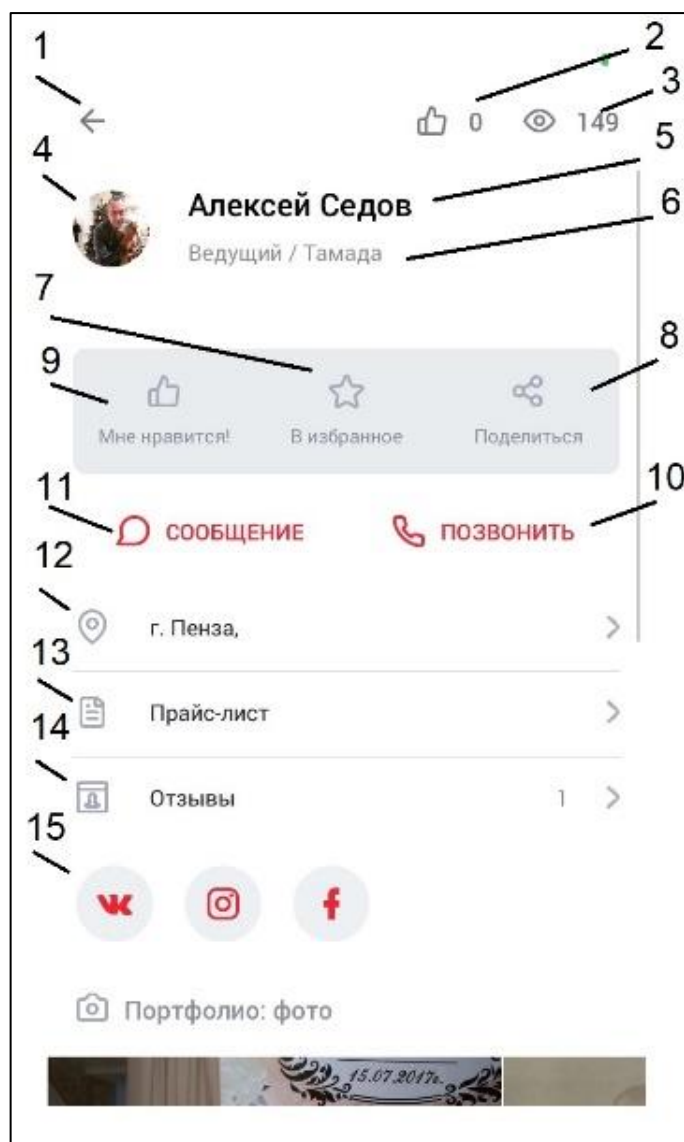


Рисунок 26 – Экран «Услуга»

Основные компоненты интерфейса экрана «Услуга»:

- 1 – кнопка возврата на предыдущий экран;
- 2 – текстовое поле, отображающее количество отметок «Мне нравится»;
- 3 – текстовое поле, отображающее количество просмотров услуги;
- 4 – аватар услуги;
- 5 – текстовое поле, отображающее название услуги;
- 6 – текстовое поле, отображающее тип услуги;
- 7 – кнопка добавления услуги в раздел «Избранное»;
- 8 – кнопка для отправки ссылки на услугу в другие приложения;
- 9 – кнопка для отметки услуги как понравившейся;

- 10 – кнопка вызова исполнителя услуги;
 - 11 – кнопка перехода к чату с исполнителем;
 - 12 – кнопка просмотра территориального расположения услуги;
 - 13 – кнопка просмотра стоимости услуги;
 - 14 – кнопка просмотра отзывов на услугу;
 - 15 – кнопки перехода к профилям в соответствующих социальных сетях.
- На рисунке 27 представлено продолжение экрана «Услуга».

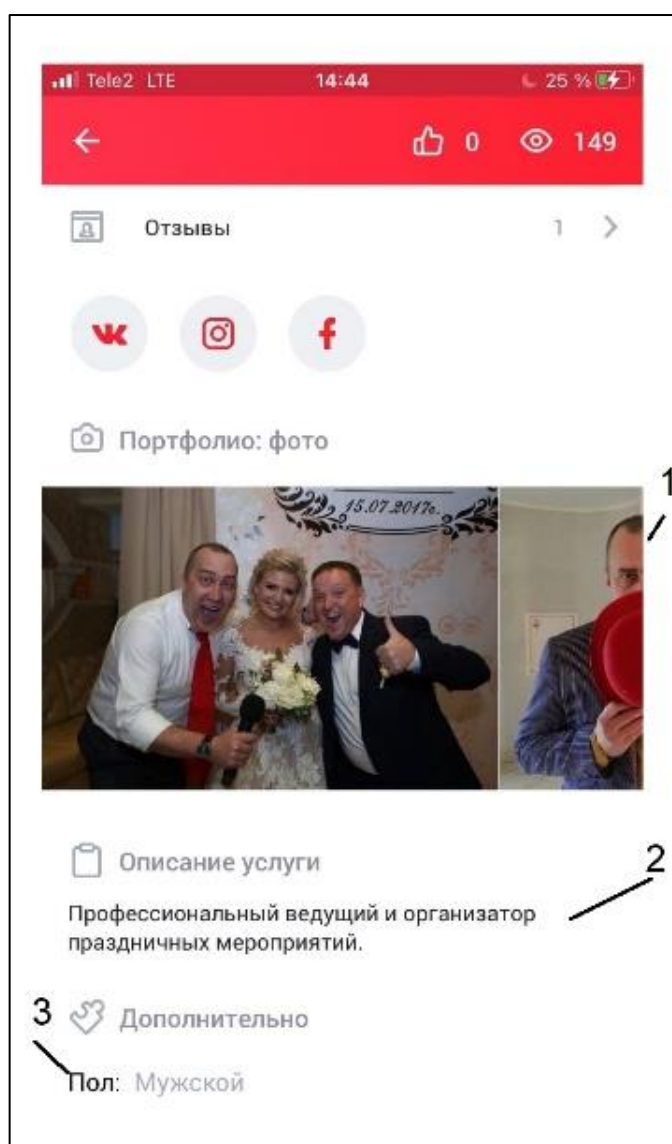


Рисунок 27 – Продолжение экрана «Услуга»

На рисунке 27:

- 1 – список картинок услуги;
- 2 – текстовое поле, отображающее описание услуги;

3 – текстовое поле, отображающее пол исполнителя.

На рисунке 28 изображен интерфейс экрана «Тендеры».

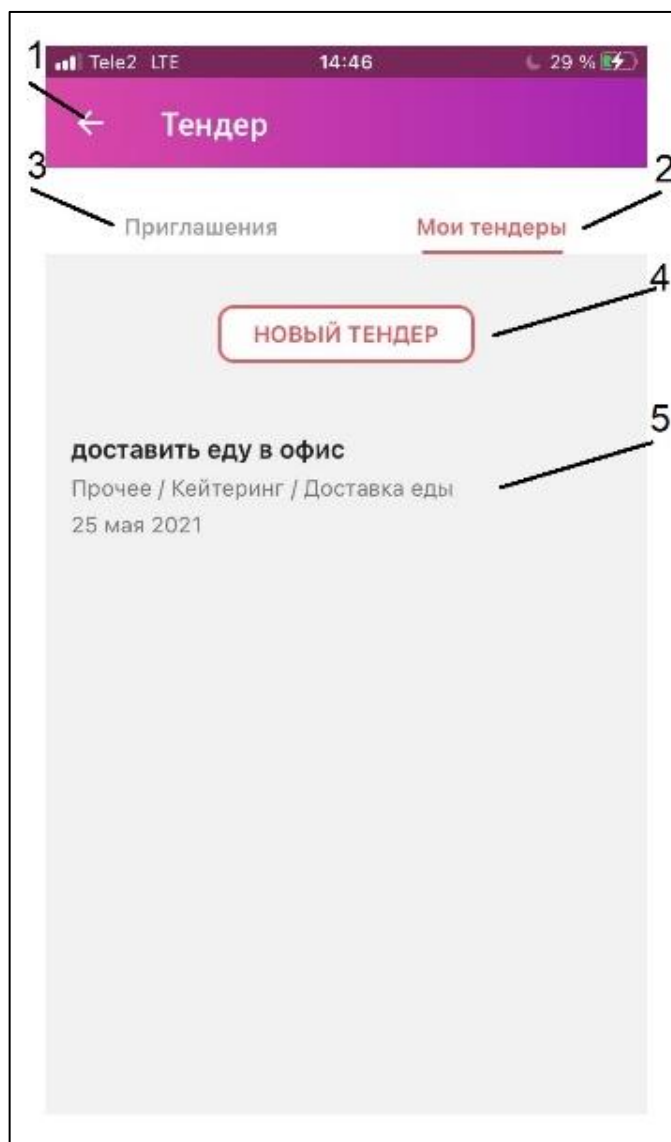


Рисунок 28 – Экран «Тендеры»

На рисунке 28:

- 1 – кнопка возврата на предыдущий экран;
- 2 – кнопка отображения раздела «Мои тендеры»;
- 3 – кнопка отображения раздела «Приглашения»;
- 4 – кнопка создания нового тендера;
- 5 – список, объявленных пользователем тендеров.

Интерфейс экрана «Создание тендера» представлен на рисунке 29.

The screenshot shows a mobile application interface for creating a new tender. The form is titled 'Новый тендер' (New Tender) and includes the following fields and controls, each indicated by a numbered callout:

- 1: Back arrow button in the top left corner.
- 2: Title bar 'Новый тендер'.
- 3: Input field for 'Название (видите только вы)' (Name (only you see)).
- 4: Dropdown menu for 'Какой будет праздник?' (Which holiday will it be?).
- 5: Dropdown menu for 'Какая нужна услуга?' (Which service is needed?).
- 6: Input field for 'Пенза' (Penza) with a dropdown arrow.
- 7: Date and time selection fields for 'Когда праздник' (When the holiday) and 'Время' (Time).
- 8: Input fields for 'Стоимость для заказчика' (Cost for the client) with 'от' (from) and 'до' (to) labels and currency symbols (₽).
- 9: Input field for 'Ваша комиссия' (Your commission) with a currency symbol (₽).
- 10: Input field for 'Дополнительная информация' (Additional information).
- 11: 'ОБЪЯВИТЬ ТЕНДЕР' (Announce Tender) button.

Рисунок 29 – Экран «Создание тендера»

Рассмотрим основные компоненты интерфейса экрана «Создание тендера»:

- 1 – кнопка возврата на предыдущий экран;
- 2 – поле ввода названия тендера;
- 3 – кнопка выбора типа праздника;
- 4 – кнопка выбора категории услуги;
- 5 – кнопка выбора города оказания услуги;
- 6 – кнопка выбора даты оказания услуги;
- 7 – поля для ввода времени оказания услуги;
- 8 – поля для ввода стоимости услуги;

- 9 – поле для ввода комиссии;
- 10 – поле для ввода дополнительно информации об услуге;
- 11 – кнопка подтверждения создания тендера.

2.4 Проектирование архитектуры приложения

Архитектура разрабатываемого приложения является клиент-серверной.

На серверной стороне реализуется база данных и предоставляется доступ к этим данным.

Клиентская сторона использует ресурсы сервера и предоставляет их пользователю с удобным графическим интерфейсом. Клиентов может быть большое количество, соответственно нужно учитывать степень нагрузки приложения как на клиенте, так и на сервере.

Клиент формирует HTTP-запрос конкретному веб-сервису. В зависимости от успешности обработки полученного от клиента запроса, веб-сервис в случае успеха отправит запрос на языке SQL серверу БД, а SQL-сервер сформирует результат выполнения и вернет запрашиваемые данные обратно веб-службе. В случае возникновения ошибок веб-сервис вернет клиенту ответ с кодом ошибки [13].

2.5 Проектирование серверной части приложения

2.5.1 Логическое проектирование базы данных

На этапе логического проектирования происходит отображение концептуальной модели предметной области в объекты используемой модели данных. Логическая модель включает в себя все сущности, атрибуты, ключи и взаимосвязи, которые представляют бизнес-информацию и определяют бизнес-правила. Была построена логическая модель базы данных в нотации IDEF1X (рисунок 30).

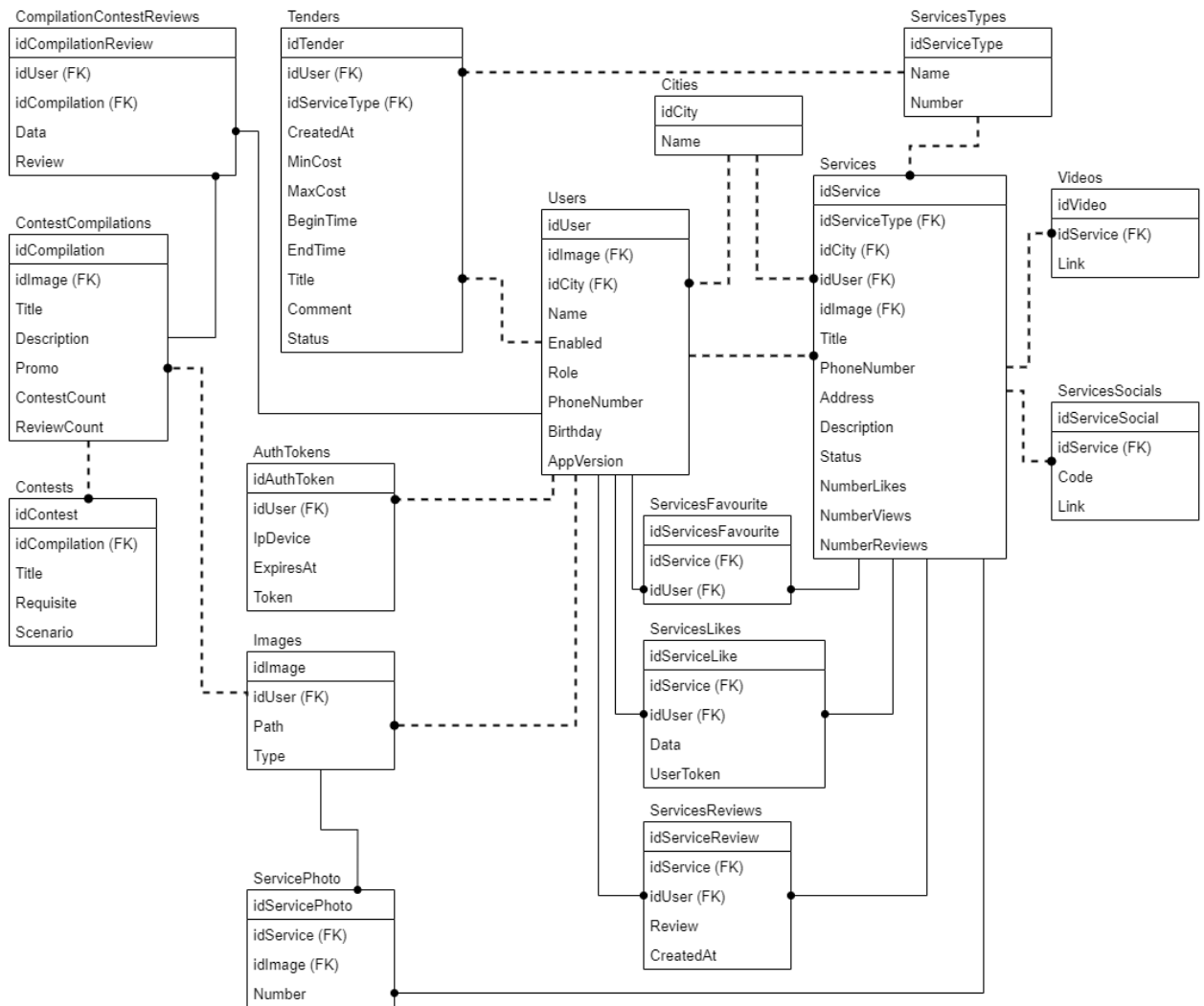


Рисунок 30 – Логическая модель базы данных

Построение проводилось с помощью онлайн сервиса для создания диаграмм «Draw.io».

2.5.2 Проектирование серверной части приложения

Серверная часть приложения представляет собой API для работы с данными, хранящимися на сервере. Передача данных осуществляется по протоколу HTTP. Формат передаваемых данных – JSON.

Написание веб-сервисов будет производиться на основе архитектурного шаблона MVC (Model-View-Controller). Для реализации API достаточно слоя Controller и Model.

Разработанная архитектура представлена на рисунке 31.

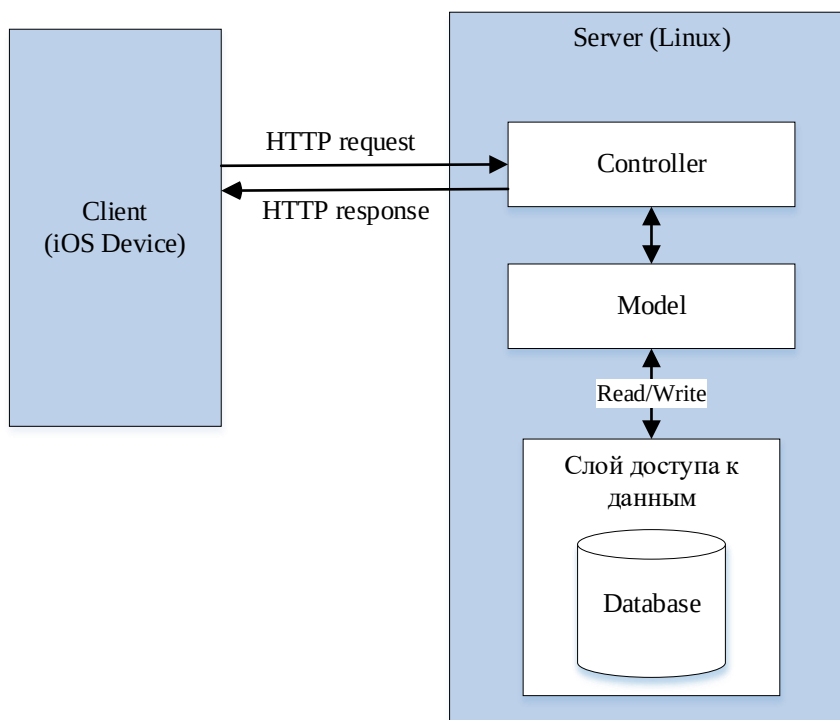


Рисунок 31 – Архитектура веб-сервиса

Веб-сервисы будут построены в архитектурном стиле REST [14]. Каждый объект определяется по уникальному URL и предоставляет 4 базовые операции для работы с данными:

- GET – получение данных;
- POST – добавление данных;
- PUT – обновление данных;
- DELETE – удаление данных.

2.6 Проектирование клиентской части приложения

При разработке клиентской части приложения принято решение использовать рекомендуемый и доработанный компанией Apple архитектурный шаблон MVC (Model-View-Controller). В архитектуру входят следующие компоненты:

- Model – компонент для работы с данными;
- View – компонент, отвечающий за отображение модели пользователю;

– Controller – компонент, обрабатывающий действия пользователя и меняющий состояние модели.

Диаграмма взаимодействия компонентов паттерна представлена на рисунке 32.

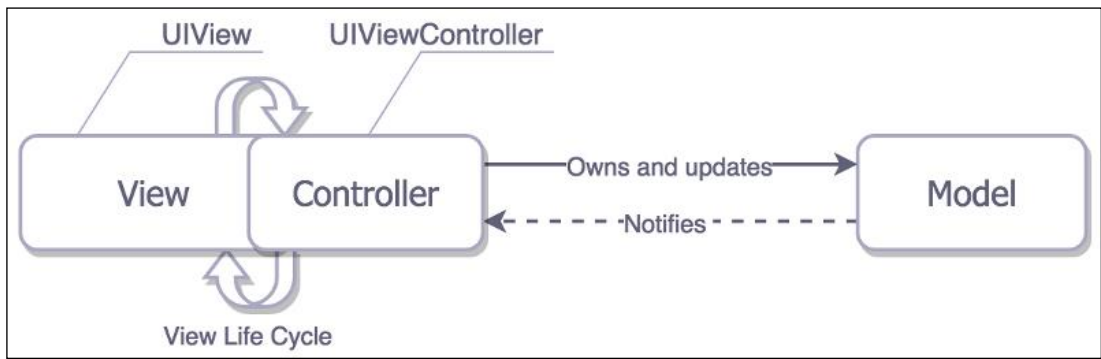


Рисунок 32 – Архитектурный шаблон MVC в реализации компании Apple

Apple поощряет написание массивных ViewController, потому что контроллер настолько вовлечен в жизненный цикл View, что трудно сказать, что он является отдельной сущностью. Есть возможность отгрузить часть бизнес-логики и преобразования данных в Model [15].

В компонент Managers клиентской части приложения будут вынесены следующие сущности:

- AccountManager – класс, управляющий состоянием пользователя в системе;
- ServerManager – класс с помощью которого будет осуществлён доступ к API сервера;
- RealmManager – класс для работы с локальной базой данных клиентского устройства;
- Validator – класс, валидирующий данные.

Также компонент Managers будет содержать глобальные для всего приложения расширения.

3 Реализация мобильного приложения

3.1 Реализация серверной части мобильного приложения

3.1.1 Реализация базы данных

На основе логической базы данных была построена физическая модель базы данных. Полученная физическая модель базы данных представлена на рисунке 32. Она включает в себя структуру базы данных, спецификацию таблиц, столбцов и ключевых ограничений.

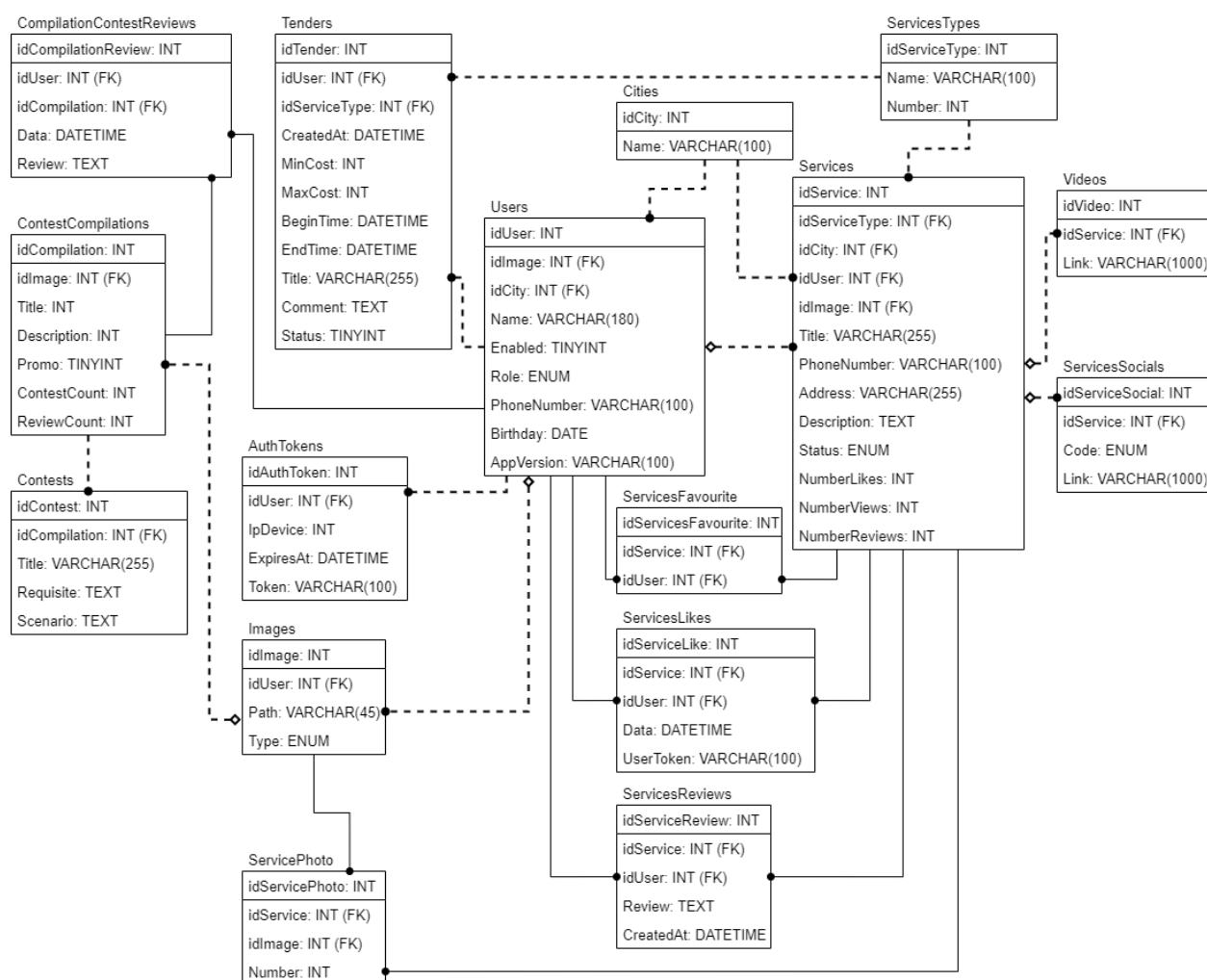


Рисунок 32 – Физическая модель базы данных

Реализована база данных с помощью СУБД MySQL Server.

3.1.2 Реализация серверной части

Разработка серверной части проводилась на языке PHP в среде разработки PhpStorm 2021.1.0 [16]. Листинг серверной части представлен в приложении Б.

3.2 Реализация клиентской части мобильного приложения

Разработка клиентской части мобильного приложения производилась на языке программирования Swift в среде Xcode. Листинг основной клиентской части мобильного приложения представлен в приложении В.

Часть графического интерфейса реализована с помощью Interface Builder – специального набора инструментов для визуализации элементов интерфейса и отдельных экранов приложения. Другая часть написана с помощью сторонней библиотеки SnapKit [17]. Она позволяет реализовать представления с помощью написания кода на Swift.

Во время реализации клиентской части также использованы следующие библиотеки:

- Just – библиотека для формирования HTTP запросов к серверному API [18];
- SideMenu – библиотека для реализации бокового меню интерфейса пользователя;
- Firebase – библиотека для сбора статистики в процессе использования приложения [19];
- Realm – библиотека для работы с локальными данными клиентского устройства [20].

Диаграмма основных компонентов клиентской части мобильного приложения представлена на рисунке 33. Она отражает общую структуру исходного кода программы и показывает взаимосвязи между компонентами. Под компонентом понимается физическая часть системы, обеспечивающая реализацию классов и отношений.

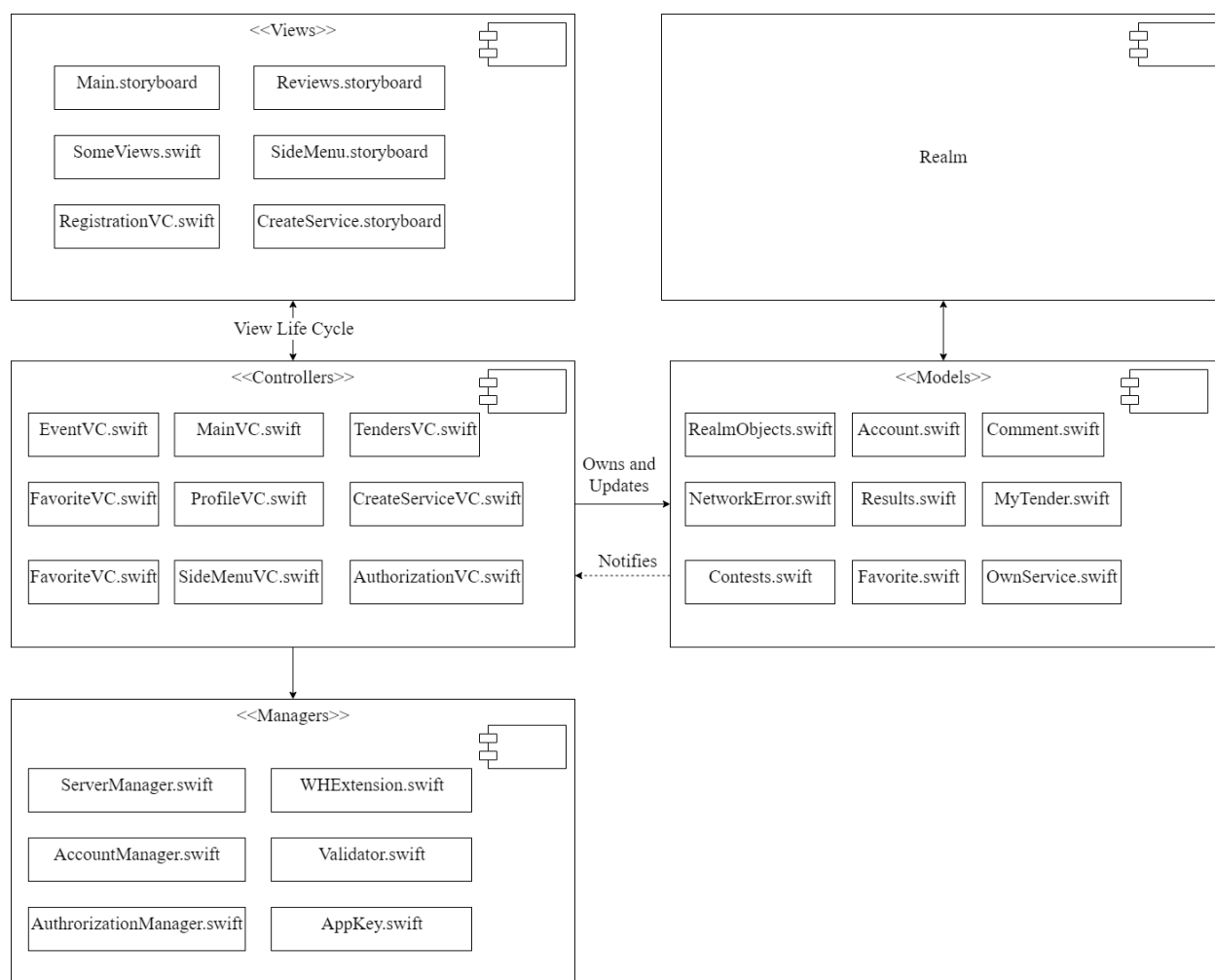


Рисунок 33 – Диаграмма компонентов мобильного приложения

На диаграмме видно, как компоненты распределены по архитектурным слоям, и как эти слои взаимодействуют друг с другом. Компонент диаграммы Realm является локальной базой данных устройства пользователя. Он служит для постоянного хранения небольшого объёма данных, актуальность которых не требует частого обновления. Это необходимо для предоставления пользователю базового функционала при отсутствии интернет соединения на устройстве.

На рисунке 34 представлена диаграмма развертывания мобильного приложения. Она отображает распределение компонентов приложения по физическим узлам и их связи на этапе исполнения. В качестве клиентов могут выступать сразу несколько различных устройств, работающих на операционной системе iOS.

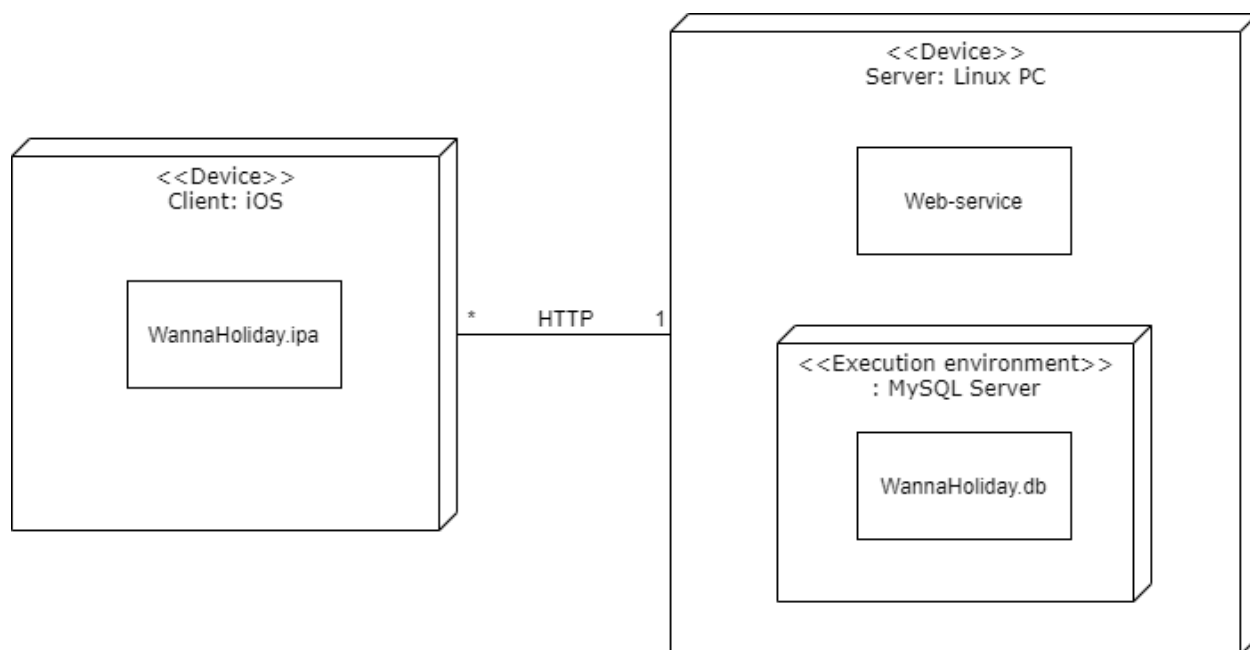


Рисунок 34 – Диаграмма развертывания мобильного приложения

На рисунке видно, что клиентская часть приложения разворачивается на устройстве пользователя, а веб-сервис и база данных разворачивается на сервере.

4 Тестирование программного продукта

4.1 Тестирование серверной части

Необходимо убедиться в корректности работы серверной части приложения. Для этого нужно протестировать каждый ресурс веб-сервиса. Тестирование запросов к API производится с помощью инструментального средства Postman [21]. Скриншоты результатов тестирования представлены в приложении Г.

Подробное описание тестов для основных запросов, которые необходимы для реализации функционала клиентской части мобильного приложения, приведено в таблице 5.

Таблица 5 – План и результаты тестирования веб-сервиса

Название ресурса	Описание теста	Запрос		Результат
		Метод	URI	
Auth	Авторизация пользователя	GET	/api/auth	Тест пройден успешно (рисунок Г.1). Отобразился id пользователя и его токен.
Cities	Получение списка городов	GET	/api/cities	Тест пройден успешно (рисунок Г.2). Получен список городов.
Events	Получение списка типов услуг	GET	/api/events	Тест пройден успешно (рисунок Г.3). Получен список конкурсов.
ContestCompilations	Получение конкурсных подборок	GET	/api/contest_compilations	Тест пройден успешно (рисунок Г.4). Получен список конкурсных подборок.
Favorites	Получение списка услуг из категории «Избранное»	GET	/api/favorites	Тест пройден успешно (рисунок Г.5). Получены избранные услуги.
	Внесение услуги с номером 3479 в категорию «Избранное»	POST	/api/favorites	Тест пройден успешно (рисунок Г.6). Код ответа – 200.

Продолжение таблицы 5

Название ресурса	Описание теста	Запрос		Результат
		Метод	URI	
	Удаление услуги с номером 3479 из категории «Избранное»	DELETE	/api/favorites	Тест пройден успешно (рисунок Г.7). Код ответа – 200.
Account	Получение данных об авторизованном пользователе	GET	/api/account	Тест пройден успешно (рисунок Г.8). Получены данные об авторизованном пользователе.
	Обновление данных об авторизовавшемся пользователе	POST	/api/account	Тест пройден успешно (рисунок Г.9). Данные о пользователе изменены.
Reviews	Получение отзывов к услуге с номером 3479	GET	/api/reviews/3479	Тест пройден успешно (рисунок Г.10). Получен список отзывов на услугу с номером 3479.
	Создание отзыва к услуге с номером 3479	POST	/api/reviews/3479	Тест пройден успешно (рисунок Г.11). Создан отзыв на услугу с номером 3479.
	Обновление отзыва к услуге с номером 3479	PUT	/api/reviews/3479	Тест пройден успешно (рисунок Г.12). Отзыв на услугу 3479 успешно обновлен.
	Удаление отзыва к услуге с номером 3479	DELETE	/api/reviews/3479	Тест пройден успешно (рисунок Г.13). Код ответа – 200.

Все проведенные тесты были пройдены успешно, что свидетельствует о корректности работы серверной части.

4.2 Тестирование клиентской части

4.2.1 Функциональное тестирование

В ходе проверки корректности работы клиентской части приложения составлен ряд функциональных тестов. Проводилось тестирование на устройстве iPhone SE (модель 2020 года выпуска) под управлением iOS.

Результаты функционального тестирования приведены в таблице 6. Скриншоты представлены в приложении Д.

Таблица 6 – План и результаты тестирования клиентской части

№ п/п	Вариант использования	Описание теста	Результаты теста
1	Просмотреть экран «Вход»	На главном экране приложения перейти к боковому меню и нажать на кнопку «Вход/регистрация».	Тест выполнен успешно. Отобразился экран «Вход» (рисунок Д.1).
2	Войти в систему	На экране «Вход» ввести ранее зарегистрированный номер телефона (9063959725) и нажать на кнопку «Позвонить мне».	Тест выполнен успешно. Отобразился экран подтверждения входа (рисунок Д.2).
3		На экране подтверждения входа ввести корректный код подтверждения и нажать на кнопку «Войти»	Тест выполнен успешно. Отобразился главный экран (рисунок Д.3).
4		На экране подтверждения входа ввести некорректный код подтверждения и нажать на кнопку «Войти»	Тест выполнен успешно. Отобразился экран с сообщением о некорректном коде подтверждения (рисунок Д.4).
5	Зарегистрироваться	На экране «Вход» ввести ранее не зарегистрированный номер телефона и нажать на кнопку «Позвонить мне».	Тест выполнен успешно. Отобразился экран «Регистрация» (рисунок Д.5).
6		На экране регистрации ввести имя «Пётр» дату рождения «02.06.1999» и город «Пенза», а также ввести код подтверждения и нажать на кнопку «Зарегистрироваться»	Тест выполнен успешно. Отобразился главный экран а в боковом меню имя зарегистрировавшегося пользователя (рисунок Д.6).
7	Просмотреть экран редактирования профиля	На главном экране приложения перейти к боковому меню и нажать на кнопку «Настройки профиля».	Тест выполнен успешно. Отобразился экран «Настройки профиля» (рисунок Д.7).

Продолжение таблицы 6

№ п/п	Вариант использования	Описание теста	Результаты теста
8	Отредактировать профиль	Выбрать картинку для профиля из галереи устройства, а также заменить имя Пётр на «Евгений» и дату рождения «02.06.1999» на «05.07.1998» и выйти с экрана редактирования	Тест выполнен успешно. В боковом меню отобразилась ранее введённая информация (рисунок Д.8).
9	Осуществить фильтрацию услуг по типу	На главном экране нажать на кнопку «Каталог исполнителей»	Тест выполнен успешно. Отобразился экран «Каталог исполнителей» с возможными типами услуг (рисунок Д.9).
10		Нажать на кнопку «Ведущий/Тамада» в списке типов услуг на экране «Каталог исполнителей»	Тест выполнен успешно. Отобразился экран «Ведущий/Тамада» с соответствующим списком (рисунок Д.10).
11	Осуществить фильтрацию услуг по цене	Нажать на кнопку «Фильтры» на экране со списком услуг и указать цену от 1000 до 2000 рублей за час. Нажать на кнопку с числом найденных исполнителей для подтверждения фильтра	Тест выполнен успешно. Отобразился список с соответствующей стоимостью услуг за час (рисунки Д.11 и Д.12).
12	Просмотреть экран услуги	Выбрать услугу «Праздничное агентство «WEEKEND»» в списке услуг категории «Ведущий/Тамада»	Тест выполнен успешно. Отобразился экран с описанием услуги «Праздничное агентство «WEEKEND»» (рисунки Д.13 и Д.14)
13	Добавить услугу в избранное	На экране описания услуги «Праздничное агентство «WEEKEND»» нажать на кнопку «В избранное»	Тест выполнен успешно. При переходе на экран «Избранное» из бокового меню в списке можно увидеть добавленную услугу (рисунок Д.15)
14	Позвонить исполнителю услуги	На экране описания услуги «Праздничное агентство «WEEKEND»» нажать на кнопку «Позвонить»	Тест выполнен успешно. Отобразилось представление с номером телефона исполнителя, по нажатию на который осуществится вызов на его мобильное устройство (рисунок Д.16)

Продолжение таблицы 6

№ п/п	Вариант использования	Описание теста	Результаты теста
15	Написать исполнителю услуги	На экране описания услуги «Праздничное агентство «WEEKEND»» нажать на кнопку «Сообщение»	Тест выполнен успешно. Произошло перенаправление пользователя в встроенное приложение iMessage для отправки сообщений исполнителю (рисунок Д.17).
16	Просмотреть отзывы к услуге	На экране описания услуги «Праздничное агентство «WEEKEND»» нажать на кнопку «Отзывы»	Тест выполнен успешно. Отобразился экран с отзывами к услуге. Так как отзывов на услугу еще нет то список пуст (рисунок Д.18).
17	Оставить отзыв к услуге	На экране отзывов к услуге «Праздничное агентство «WEEKEND»» написать отзыв «Хорошая услуга» в поле «Ваш отзыв» и нажать на кнопку «Опубликовать»	Тест выполнен успешно. Добавленный отзыв отобразился на экране (рисунок Д.19).
18	Просмотреть список созданных услуг	На экране «Профиль» выбрать пункт «Ваши услуги»	Тест выполнен. Отобразился экран с списком созданных услуг (рисунок Д.20).
19	Создать услугу	На экране с списком услуг нажать на кнопку «Добавить услугу»	Тест выполнен успешно. Отобразился экран с списком категорий (рисунок Д.21).
20		На экране с списком категорий выбрать категорию «Dj/ Звуковое оформление»	Тест выполнен успешно. Отобразился экран создания услуги для категории «Dj/ Звуковое оформление» (рисунок Д.22).
21		В поле «Описание услуги» вписать «Предоставление музыкальной аппаратуры», добавить картинку услуги и фото портфолио из галереи устройства. Также вписать ссылку на видео « https://www.youtube.com/watch?v=9lWuFLCozSw&t=888s » и цену за час «2000». Нажать кнопку «Опубликовать»	Тест выполнен успешно. В списке созданных услуг отобразилась созданная ранее услуга (рисунок Д.23).
22	Создать тендер	На главном экране нажать на кнопку «Тендеры»	Тест выполнен успешно. Отобразился экран «Тендеры» (рисунок Д.24).
23		Выбрать пункт мои тендеры и нажать на кнопку «Новый тендер»	Тест выполнен успешно. Отобразился экран создания тендера (рисунок Д.25).

Окончание таблицы 6

№ п/п	Вариант использования	Описание теста	Результаты теста
24		Выбрать тип праздника «Детский праздник» и категорию «Ведущий/Тамада», указать дату «05.06.2021» и время с «15:00» до «18:00», стоимость от 1000 до 3000 рублей. Нажать на кнопку «Объявить тендер».	Тест выполнен успешно. На экране «Мои тендеры» отобразился созданный ранее тендер (рисунок Д.26).
25	Просмотреть список подборок конкурсов	Выбрать пункт «Конкурсы» в боковом меню	Тест выполнен успешно. Отобразился список подборок с конкурсами (рисунок Д.27).
26	Просмотреть подборку конкурсов	Выбрать пункт «Эстафеты для компании» в списке подборок конкурсов	Тест выполнен успешно. Отобразился экран с информацией о подборке «Эстафеты для компании» (рисунок Д.28).
27	Купить подборку конкурсов	Нажать на кнопку «купить за 249 р» на экране подборки конкурсов «Эстафеты для компании». Пройти последующую процедуру оплаты подборки.	Тест выполнен успешно. На экране подборки «Эстафеты для компании» отобразился список с конкурсами (рисунки Д.29 и Д.30).

Функциональные тесты пройдены успешно, что свидетельствует о корректности работы клиентской части мобильного приложения.

4.2.2 Модульное тестирование

Модульный тест позволяет вызвать автономную единицу работы и проверить предположения о конечном результате этой единицы. Данный вид тестирования облегчает обнаружение и устранение ошибок после изменения кода.

Разработан тестовый класс wannaholidayTests, содержащий 7 unit-тестов, описание которых приведено в таблице 7. Модульное тестирование проводилось с помощью фреймворка XCTest от компании Apple [22].

Таблица 7 – Описание unit-тестов

Название тестового метода	Описание теста
testPhoneValidateWithCorrectPhone	Тестирование метода валидации номера телефона с корректными данными.
testPhoneValidateWithIncorrectLongPhone	Тестирование метода валидации номера телефона превышающего допустимую длину.
testPhoneValidateWithIncorrectSmallPhone	Тестирование метода валидации номера телефона с малой длиной.
testEmailWithCorrectData	Тестирование метода валидации почтового адреса с корректными данными.
testEmailWithIncorrectData	Тестирование метода валидации почтового адреса с некорректными данными.
testSmsCodeWithCorrectData	Тестирование метода валидации sms кода с корректными данными.
testSmsCodeWithIncorrectData	Тестирование метода валидации sms кода с некорректными данными.

Результаты тестирования представлены на рисунке 35.

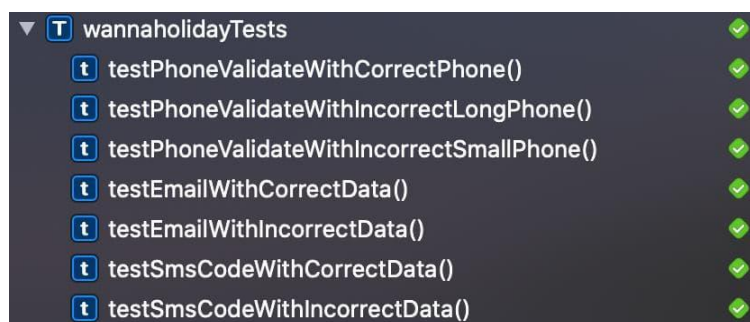


Рисунок 35 – Результаты выполнения модульных тестов

На рисунке представлен список всех тестов и их статус выполнения. Все unit-тесты были успешно пройдены.

4.2.3 Метрики кода

Для оценки уровня качества разработанного мобильного приложения подсчитаем метрики кода, т.е. получим количественные значения определенных свойств программного обеспечения [23].

Часто используемой метрикой является Maintainability Index (MI) – это комплексный показатель качества кода, который рассчитывается по следующей формуле [24]:

$$MI = \max\left(0, (171 - 5.2 * \ln(HV) - 0.23 * CC - 16.2 * \ln(LoC)) * \frac{100}{171}\right),$$

где NV – вычислительная сложность (HalsteadVolume);

CC – показатель структурной сложности кода, или количество различных ветвей в коде (CyclomaticComplexity);

LoC – количество строк кода.

Данная метрика принимает значения от 0 до 100, чем больше значение, тем проще поддерживать код [24].

Для начала необходимо рассчитать вычислительную сложность NV по следующей формуле:

$$NV = N \log_2 \eta,$$

где N – длина программы;

η – словарь программы.

Длина программы рассчитывается по формуле:

$$N = N_1 + N_2, \quad (2)$$

где N_1 – общее число операторов в программе;

N_2 – общее число операндов в программе.

Словарь программы:

$$\eta = \eta_1 + \eta_2,$$

где η_1 – число уникальных операторов в программе, включая символы-разделители, имена процедур и знаки операций;

η_2 – число уникальных операндов в программе [25].

Показатель структурной сложности кода CC рассчитывается по следующей формуле:

$$CC = KOP + KOЦ + 1,$$

где KOP – количество операторов разветвления;

$KOЦ$ – количество операторов цикла.

Рассчитаем данные показатели для метода анимации окна с отзывом на подборку конкурсов `animateCommentView()` класса `CompilationReviewsVC`, представленного в листинге 1.

Листинг 1 – Анимация окна с отзывом на подборку конкурсов

```
// MARK: - Animations
// Анимация открытия/закрытия панели отзыва пользователя
private func animateCommentView() {
    UIView.animate(withDuration: 0.2, animations: {
        if self.commentViewIsVisible {
            self.userCommentView.transform = CGAffineTransform(translationX: 0, y: 122)
            self.commentViewButton.setImage(UIImage(named: "upArrowIcon"), for: .normal)
            self.commentViewIsVisible = false
            self.bottomTableViewConstraint.constant = 40
        } else {
            self.userCommentView.transform = CGAffineTransform(translationX: 0, y: 0)
            self.commentViewButton.setImage(UIImage(named: "downArrowIcon"), for: .normal)
            self.commentViewIsVisible = true
            DispatchQueue.main.asyncAfter(deadline: .now() + 0.2, execute: {
                self.bottomTableViewConstraint.constant = 162
            })
        }
    })
}
```

Данные, необходимые для расчетов словаря и длины программы представлены в таблице 8.

Таблица 8 – Данные для расчета словаря и длины программы

Операторы			Операнды		
№	Оператор	Количество	№	Операнд	Количество
1	()	10	1	commentViewIsVisible	3
2	{}	5	2	userCommentView	2
3	=	6	3	CGAffineTransform	2
4	+	1	4	false	1
5	if	1	5	true	1
6	,	6	6	commentViewButton	2
			7	bottomTableViewConstraint	2
			8	main	1
			9	UIView	1
			10	UIImage	2
Всего:	$\eta_1 = 6$	$N_1 = 29$	Всего:	$\eta_2 = 10$	$N_2 = 17$

Словарь программы $\eta = 6 + 10 = 16$.

Длина программы $N = 33 + 10 = 46$.

Вычислительная сложность $HV = 46 * 4 = 184$.

Количество строк кода LoC = 19.

Структурная сложность $CC = 1 + 0 + 1 = 2$.

Теперь можно рассчитать Maintainability Index:

$$MI = \max\left(0, (171 - 5.2 * \ln(184) - 0.23 * 2 - 16.2 * \ln(19)) * \frac{100}{171}\right) = 56.02.$$

На основе полученных результатов можно сделать вывод, что разработанный фрагмент кода имеет достаточно высокий уровень качества, т.к. значение комплексного показателя составляет 56.02. В дальнейшем такой код будет легко поддерживать.

Заключение

В ходе выполнения выпускной квалификационной работы было разработано мобильное приложение поддержки процессов взаимного поиска потребителей и поставщиков социальных услуг.

На этапе анализа предметной области и требований были рассмотрены существующие аналоги на рынке мобильных приложений, выделены их преимущества и недостатки, на основе чего были выработаны функциональные и нефункциональные требования к разрабатываемому продукту.

На этапе проектирования было осуществлено планирование разработки мобильного приложения и оценка бюджета. Таким образом, длительность проекта составляет 122 дня, а затраты на реализацию – 771600 рублей. Также была разработана логическая модель базы данных и спроектирована архитектура клиентской и серверной части.

Мобильное приложение было реализовано на языке Swift в среде разработки Xcode с использованием шаблона проектирования MVC. Для реализации базы данных использовалась СУБД MySQL Server и язык SQL.

В ходе выполнения выпускной квалификационной работы было проведено тестирование серверной части, функциональное и модульное тестирование клиентской части, на основе которого был сделан вывод о корректности работы всего приложения.

Разработанный продукт успешно опубликован в магазине iOS приложений AppStore. Акт о внедрении мобильного приложения представлен в приложении Е.

Таким образом, в результате выполнения выпускной квалификационной работы было разработано мобильное приложение поддержки процессов взаимного поиска потребителей и поставщиков социальных услуг, отвечающее всем требованиям технического задания.

Список использованных источников

1. Распоряжение Правительства Российской Федерации от 28 июля 2017 г. № 1632-р Программа «Цифровая экономика Российской Федерации» – URL:
<http://static.government.ru/media/files/9gFM4FHj4PsB79I5v7yLVuPgu4bvR7M0.pdf> (дата обращения: 21.05.2018)
2. Распоряжение Президента Российской Федерации от 9 мая 2017 г. № 203 «Стратегия развития информационного общества в Российской Федерации на 2017 – 2030 годы – URL:
http://zmedu.hostedu.ru/files/ykaz_7668.pdf (дата обращения: 21.05.2018)
3. Григорьева, Д.Р. Применение информационных технологий в социальной сфере / Д.Р. Григорьева, А.С. Попченко. – Молодой учёный. – 2015. – №11 – С. 177-179.
4. Свадебный эксперт – свадьба и корпоративы в одном приложении // App Store [Электронный ресурс]. – <https://apps.apple.com/ru/app/свадебный-эксперт-свадьба-и-корпоративы-в-одном-приложении/id899110980> (дата обращения: 20.05.2021)
5. Мой Профи: услуги рядом с вами // App Store [Электронный ресурс]. – <https://apps.apple.com/ru/app/мой-профи-услуги-рядом-с-вами/id975558645> (дата обращения: 20.05.2021)
6. Яндекс.Услуги – уборка, ремонт // App Store [Электронный ресурс]. – <https://apps.apple.com/ru/app/яндекс-услуги-уборка-ремонт/id1559549371> (дата обращения: 20.05.2021)
7. Яндекс.Услуги для мастеров // App Store [Электронный ресурс]. – <https://apps.apple.com/ru/app/яндекс-услуги-для-мастеров/id1537579597> (дата обращения: 20.05.2021)
8. Арлоу, Д. UML 2 и Унифицированный процесс. Практический объектно-ориентированный анализ и проектирование, 2е издание / Д. Арлоу, И. Нейштадт – СПб: Символ Плюс, 2007. – 624 с.

9. Методология. Стандарт IDEF1 // Iteranet IT-журнал [Электронный ресурс]. – URL: <http://iteranet.ru/it-novosti/2013/10/16/metodologiya-standart-idef1/> (дата обращения: 22.05.2021)
10. Хаф Л. Методология разработки программного обеспечения // Компьютер Пресс [Электронный ресурс]. – URL: <https://compress.ru/article.aspx?id=9633> (дата обращения: 19.05.2021)
11. Богданов, В.В. Управление проектами в Microsoft Project 2007. Учебный курс (+CD) / В.В. Богданов – Спб.: Питер, 2008. – 592 с.
12. Human Interface Guidelines // developer.apple.com [Электронный ресурс]. – URL: <https://developer.apple.com/design/human-interface-guidelines/> (дата обращения: 02.06.2021)
13. Грамотная клиент-серверная архитектура: как правильно проектировать и разрабатывать web API // tproger.ru [Электронный ресурс]. – URL: <https://tproger.ru/articles/web-api/> (Дата обращения: 03.06.2021)
14. Архитектура REST // habr.com [электронный ресурс]. – URL: <https://habr.com/ru/post/38730/> (дата обращения: 03.06.2021)
15. Блог компании Badoo. – Архитектурные паттерны в iOS // habr.com [электронный ресурс]. – URL: <https://habr.com/ru/company/badoo/blog/281162/> (дата обращения: 04.06.2021)
16. Документация и демо – PhpStorm // www.jetbrains.com [электронный ресурс]. – URL: <https://www.jetbrains.com/ru-ru/phpstorm/documentation/> (дата обращения: 04.06.2021)
17. SnapKit // github.com [электронный ресурс]. – URL: <https://github.com/SnapKit/SnapKit> (дата обращения: 05.06.2021)
18. Just – Swift HTTP for Humans // github.com [электронный ресурс]. – URL: <https://github.com/dduan/Just> (дата обращения: 05.06.2021)
19. Add Firebase to your iOS project // firebase.google.com [электронный ресурс]. – URL: <https://firebase.google.com/docs/ios/setup?hl=nl> (дата обращения: 05.06.2021)

20. RealmSwift Reference // docs.mongodb.com [электронный ресурс]. – URL: <https://docs.mongodb.com/realm-sdks/swift/latest/> (дата обращения: 05.06.2021)
21. Create API Documentation with Postman // www.postman.com [электронный ресурс]. – URL: <https://www.postman.com/api-documentation-tool/> (дата обращения: 08.06.2021)
22. XCTest Framework // developer.apple.com [электронный ресурс]. – URL: <https://developer.apple.com/documentation/xctest> (дата обращения: 08.06.2021)
23. Рязжков Е. Программный код и его метрики // Habr [Электронный ресурс]. – URL: <https://habr.com/company/intel/blog/106082/> (дата обращения: 31.05.2018)
24. Средство оценки качества кода // habr.com [электронный ресурс]. – URL: <https://habr.com/ru/post/111524/> (09.06.2021)

Приложение А
ГЛОССАРИЙ
(справочное)

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

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

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

SQL (Structured Query Language) – декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных.

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.

REST (Representational State Transfer) – архитектурный стиль взаимодействия компонентов распределённого приложения в сети.

Xcode – интегрированная среда разработки программного обеспечения, разработанная корпорацией Apple для своих платформ.

Приложение Б
ИСХОДНЫЙ КОД СЕРВЕРНОЙ ЧАСТИ
(рекомендуемое)

Листинг Б.1 – ServiceProviderController.php

```

<?php

declare(strict_types=1);

namespace ApiBundle\Controller;

use AppBundle\Entity\User;
use AppBundle\Services\DTOs\AppServiceProfiles\CreateEditServiceDTO;
use AppBundle\Entity\AppServicesProfiles;
use AppBundle\Entity\AppServicesWidgets;
use AppBundle\Services\AppServicesProfilesService;
use AppBundle\Services\AppServiceWidgetService;
use AppBundle\Transformers\AppServiceProfilesTransformer;
use phpDocumentor\Reflection\DocBlock\Tags\Uses;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\HttpFoundation\JsonResponse;
use Doctrine\ORM\EntityManagerInterface;
use Symfony\Component\Routing\Annotation\Route;

class ServiceProviderController extends BaseController
{
    /**
     * @var AppServicesProfilesService
     */
    private $appServicesProfilesService;
    /**
     * @var EntityManagerInterface
     */
    private $entityManager;
    /**
     * @var AppServiceProfilesTransformer
     */
    private $profilesTransformer;

    public function __construct(
        AppServicesProfilesService $appServicesProfilesService,
        EntityManagerInterface $entityManager,
        AppServiceProfilesTransformer $profilesTransformer
    ) {
        $this->appServicesProfilesService =
$AppServicesProfilesService;
        $this->entityManager = $entityManager;
        $this->profilesTransformer = $profilesTransformer;
    }

    /**
     * Возвращает список всех услуг, созданных пользователем.
     * @Route("/service_provider/services", methods={"GET"})
     * @param Request $request
     * @return JsonResponse

```

```

    */
    public function servicesListAction(Request $request)
    {
        $query = $request->get('query');
        $offset = (int)$request->get('offset');
        $limit = (int)$request->get('limit');
        /** @var User $user */
        $user = $this->getUser();

        $servicesList = [];
        $servicesProfiles = $this->appServicesProfilesService-
>getRepository()->findByUserAndPaginator(
            $user,
            $query,
            $offset,
            $limit
        );
        foreach ($servicesProfiles as $servicesProfile) {
            $servicesList[] = [
                'id' => $servicesProfile->getId(),
                'service_type_id' => $servicesProfile-
>getServiceType()->getId(),
                'events' => $servicesProfile-
>getEventsWithHiddenEvents()->map(
                    function (AppServicesWidgets $widgets) {
                        return $widgets->getId();
                    }
                ),
                'city_id' => $servicesProfile->getCity()-
>getId(),
                'name' => $servicesProfile->getDisplayName(),
                'status' => $servicesProfile->getStatus(),
                'enhancements' => $this->profilesTransformer-
>getActiveEnhancements($servicesProfile),
            ];
        }

        return $this->json($servicesList);
    }

    /**
     * Возвращает данные об указанной услуге, подлежащие
     редактированию исполнителем.
     * @Route("/service_provider/services/{id}" ,
methods={"GET"})
     * @param int $id
     * @param AppServiceWidgetService $appServiceWidgetService
     * @return JsonResponse
     */
    public function servicesViewAction(
        int $id,
        AppServiceWidgetService $appServiceWidgetService
    ) {

```

```

        $AppServicesProfiles = $this->
>appServicesProfilesService->getRepository()->get($id);
        $params = $appServiceWidgetService->
>getServiceParams($AppServicesProfiles);

        $socialsArray = $AppServicesProfiles->getSocialsArray();

        return $this->json(
            [
                'serviceTypeId' => $AppServicesProfiles->
>getServiceType()->getId(),
                'status' => $AppServicesProfiles->getStatus(),
                'avatarUrl' => $AppServicesProfiles->
>getAvatarImageArray(),
                'displayName' => $AppServicesProfiles->
>getDisplayName(),
                'contactPhoneNumber' => $AppServicesProfiles->
>getPhoneNumber(),
                'cityId' => $AppServicesProfiles->getCity()->
>getId(),
                'address' => $AppServicesProfiles->getAddress(),
                'description' => $AppServicesProfiles->
>getDescription(),
                'portfolioPhotos' => $AppServicesProfiles->
>getPhotosArray(),
                'portfolioVideos' => $AppServicesProfiles->
>getVideosArray(),
                'socialLinks' => sizeof($socialsArray) ?
                $socialsArray : null,
                'enhancements' => $this->profilesTransformer->
>getActiveEnhancements($AppServicesProfiles),
                'params' => $params,
            ]
        );
    }

    /**
     * Создаёт новую услугу, принадлежащую вызвавшему метод
     * пользователю.
     * Если запрос совершён пользователем, имеющим статус
     * "Заказчик",
     * сервер должен изменить статус пользователя на
     * "Исполнитель".
     * Услуга может создаваться с различным статусом
     * (указывается клиентом):draft и review
     *
     * @Route("/service_provider/services", methods={"POST"})
     * @param Request $request
     * @param CreateEditServiceDTO $DTO
     * @return JsonResponse
     */
    public function servicesCreateAction(Request $request,
        CreateEditServiceDTO $DTO)

```

```

    {
        $data = json_decode((string)$request->getContent(),
true);
        $DTO->importFromArray($data);
        /** @var User $user */
        $user = $this->getUser();

        $appServiceProfile = $this->appServicesProfilesService-
>create($DTO, $user);

        $this->entityManager->flush();

        return $this->json(['id' => $appServiceProfile-
>getId()], JsonResponse::HTTP_CREATED);
    }

    /**
     * Изменяет данные существующей услуги, принадлежащей
     * вызвавшему метод исполнителю.
     * @Route("/service_provider/services/{id}" ,
methods={"PUT"})
     * @param int $id
     * @param Request $request
     * @param CreateEditServiceDTO $DTO
     * @return JsonResponse
     */
    public function servicesEditAction(int $id, Request
$request, CreateEditServiceDTO $DTO)
    {
        $data = json_decode((string)$request->getContent(),
true);
        $DTO->importFromArray($data);
        /** @var User $user */
        $user = $this->getUser();

        $appServiceProfile = $this->appServicesProfilesService-
>edit($id, $DTO, $user);

        $this->entityManager->flush();
        return $this->json(['id' => $appServiceProfile-
>getId()], JsonResponse::HTTP_OK);
    }

    /**
     * Удаляет указанную услугу.
     * @Route("/service_provider/services/{id}" ,
methods={"DELETE"})
     * @param int $id
     * @return JsonResponse
     */
    public function serviceDeleteAction(int $id)
    {
        /** @var User $user */

```

```

        $user = $this->getUser();
        $this->appServicesProfilesService->delete($id, $user);

        $this->entityManager->flush();
        return $this->json([], JsonResponse::HTTP_OK);
    }
}

```

Листинг Б.2 –AppServicesProfilesService.php

```

<?php

declare(strict_types=1);

namespace AppBundle\Services;

use AppBundle\Entity\AppServicesProfiles;
use AppBundle\Entity\AppServicesProfilesValues;
use AppBundle\Entity\User;
use AppBundle\Exceptions\AccessDeniedDomainException;
use AppBundle\Repository\AppServicesProfilesRepository;
use
AppBundle\Services\DTOs\AppServiceProfiles\CreateEditServiceDTO;
use AppBundle\Widgets\Conveyor\WidgetConveyor;
use AppBundle\Widgets\Factory\Listeners\DBFillListener;
use
AppBundle\Widgets\Factory\Listeners\DescForSimplePriceListener;
use AppBundle\Widgets\Factory\SystemWidget;
use AppBundle\Widgets\Fillers\DB\DBFillerFactory;
use AppBundle\Widgets\PriceSimpleDescrWidget;
use
AppBundle\Widgets\Views\AppServiceWidget\UserServiceParams\UserS
ervicParamsFactory;
use
AppBundle\Widgets\Views\AppServiceWidget\UserServicePriceInfo\Us
erServicePriceInfoFactory;
use
AppBundle\Widgets\Views\Widget\MinPrice\MinPriceViewSysFactory;

class AppServicesProfilesService
{
    /** @var AppServicesProfilesRepository */
    protected $appServicesProfilesRepository;
    /**
     * @var AppProfileSetDataService
     */
    private $appProfileSetDataService;
    /**
     * @var AppServiceProfileVideoService
     */
    private $appServiceProfileVideoService;
    /**

```

```

    * @var ImageService
    */
private $imageService;
/**
    * @var DBFillListener
    */
private $DBFillListener;
/**
    * @var UserServicePriceInfoFactory
    */
private $priceInfoFactory;
/**
    * @var DescForSimplePriceListener
    */
private $descForSimplePriceListener;
/**
    * @var UserServicParamsFactory
    */
private $paramsFactory;
/**
    * @var WidgetConveyor
    */
private $widgetConveyor;
/**
    * @var DBFillerFactory
    */
private $DBFillerFactory;
/**
    * @var MinPriceViewSysFactory
    */
private $minPriceViewSysFactory;

public function __construct(
    AppServicesProfilesRepository
$appServicesProfilesRepository,
    AppProfileSetDataService $appProfileSetDataService,
    AppServiceProfileVideoService
$appServiceProfileVideoService,
    ImageService $imageService,
    DBFillListener $DBFillListener,
    MinPriceViewSysFactory $minPriceViewSysFactory,
    UserServicePriceInfoFactory $priceInfoFactory,
    DescForSimplePriceListener $descForSimplePriceListener,
    UserServicParamsFactory $paramsFactory,
    WidgetConveyor $widgetConveyor,
    DBFillerFactory $DBFillerFactory
) {
    $this->appServicesProfilesRepository =
$appServicesProfilesRepository;
    $this->AppProfileSetDataService =
$appProfileSetDataService;

```

```

        $this->appServiceProfileVideoService =
$appServiceProfileVideoService;
        $this->imageService = $imageService;
        $this->DBFillListener = $DBFillListener;
        $this->priceInfoFactory = $priceInfoFactory;
        $this->descForSimplePriceListener =
$descForSimplePriceListener;
        $this->paramsFactory = $paramsFactory;
        $this->widgetConveyor = $widgetConveyor;
        $this->DBFillerFactory = $DBFillerFactory;
        $this->minPriceViewSysFactory = $minPriceViewSysFactory;
    }

    public function get(int $id): AppServicesProfiles
    {
        return $this->appServicesProfilesRepository->get($id);
    }

    /**
     * @param int $id
     * @return AppServicesProfiles|null
     */
    public function findById(int $id): ?AppServicesProfiles
    {
        return $this->appServicesProfilesRepository->find($id);
    }

    public function incrementTotalViews(AppServicesProfiles
$profileData): self
    {
        $profileData->incrementTotalViews();

        $this->appServicesProfilesRepository-
>save($profileData);

        return $this;
    }

    /**
     * @return AppServicesProfiles[]
     */
    public function getAll()
    {
        return $this->appServicesProfilesRepository->getAll();
    }

    public function getRepository():
AppServicesProfilesRepository
    {
        return $this->appServicesProfilesRepository;
    }

    /**

```

```

        * Создание услуги пользователем. Пользователь становится
        исполнителем
        */
        public function create(CreateEditServiceDTO $DTO, User
        $user): AppServicesProfiles
        {
            $DTO->validate();
            $appServicesProfiles = new AppServicesProfiles();

            $this->AppProfileSetDataService->
            >setDataToService($appServicesProfiles, $DTO, $user);
            $user->setType(User::EXECUTOR_TYPE);

            $this->appServicesProfilesRepository->
            >persist($appServicesProfiles);
            return $appServicesProfiles;
        }

        /**
        * Редактирование услуги пользователем
        * @param int $id
        * @param CreateEditServiceDTO $DTO
        * @param User $user
        * @return AppServicesProfiles
        * @throws AccessDeniedDomainException
        * @throws \AppBundle\Exceptions\ValidationDomainException
        */
        public function edit(int $id, CreateEditServiceDTO $DTO,
        User $user): AppServicesProfiles
        {
            $DTO->validate();
            $appServicesProfiles = $this->
            >appServicesProfilesRepository->get($id);

            if ($appServicesProfiles->getUser() !== $user) {
                throw new AccessDeniedDomainException('Нельзя
                редактировать чужую услугу');
            }
            $this->AppProfileSetDataService->
            >setDataToService($appServicesProfiles, $DTO, $user);
            return $appServicesProfiles;
        }

        /**
        * Удаление услуги пользователем
        */
        public function delete(int $id, User $user): void
        {
            $appServicesProfiles = $this->
            >appServicesProfilesRepository->get($id);
            if ($appServicesProfiles->getUser() !== $user) {
                throw new AccessDeniedDomainException('Нельзя
                удалять чужую услугу');
            }
        }
    }

```

```

    }
    $this->appServiceProfileVideoService-
>deletePreviewsByAppService($appServicesProfiles);
    $image = $appServicesProfiles->getImage();
    if ($image !== null) {
        $this->imageService->delete($image);
    }
    $this->imageService-
>deleteServicePhotos($appServicesProfiles);
    $this->getRepository()->remove($appServicesProfiles);
}

/**
 * Поиск минимальной цены для услуги
 */
public function getMinimalPrice(AppServicesProfiles
$profileData): string
{
    $priceArray = [];

    /** @var AppServicesProfilesValues $value */
    foreach ($profileData->getValues() as $value) {
        $sysWidget =
SystemWidget::createFromAppServiceWidget($value->getWidget());
        $price = $this->widgetConveyor
            ->fillAndRender(
                $sysWidget,
                $this->DBFillerFactory,
                $this->minPriceViewSysFactory,
                (string)$value->getValue()
            );
        if (is_numeric($price)) {
            $priceArray[] = $price;
        }
    }
    if (count($priceArray) > 0) {
        $price = min($priceArray);
        return sprintf('от %d ₺', $price);
    } else {
        return '';
    }
}

public function getParams(AppServicesProfiles $profiles):
array
{
    /** @var AppServicesProfilesValues[] $appValues */
    $appValues = $profiles->getValues();
    $params = [];
    $this->widgetConveyor->setCityId($profiles->getCity()-
>getId());
    foreach ($appValues as $value) {

```

```

        $sysWidget =
SystemWidget::createFromAppServiceWidget($value->getWidget());
        $param = $this->widgetConveyor
            ->fillAndRender($sysWidget, $this-
>DBFillerFactory, $this->paramsFactory, (string)$value-
>getValue());
        if ($param) {
            $params[] = $param;
        }
    }
    return $params;
}

/**
 * Создает разметку для экрана описания цен
 * @param AppServicesProfiles $profiles
 * @return array
 */
public function getPriceInfo(AppServicesProfiles $profiles):
array
{
    /** @var AppServicesProfilesValues[] $appValues */
    $appValues = $profiles->getValues();
    $prices = [];
    $creator = $this->widgetConveyor->getWidgetFactory();

    // Получаем значение PriceSimpleDescrWidget если он
    есть, чтобы заменить описание цены для PriceSimpleWidget
    $simplePriceText = null;
    foreach ($appValues as $value) {
        $sysWidget =
SystemWidget::createFromAppServiceWidget($value->getWidget());
        $widget = $creator->create($sysWidget);
        if ($widget instanceof PriceSimpleDescrWidget) {
            /** @var PriceSimpleDescrWidget
$filledDescWidget */
            $filledDescWidget = $this->DBFillerFactory-
>create($widget)->fill((string)$value->getValue());
            $simplePriceText = $filledDescWidget->getText();
        }
    }

    foreach ($appValues as $value) {
        $sysWidget =
SystemWidget::createFromAppServiceWidget($value->getWidget());
        $price = $this->widgetConveyor
            ->fillAndRender(
                $sysWidget,
                $this->DBFillerFactory,
                $this->priceInfoFactory,
                (string)$value->getValue(),
                ['simplePriceText' => $simplePriceText]
            );
    }
}

```

```
        if ($price) {  
            $prices[] = $price;  
        }  
    }  
    return $prices;  
}  
}
```

Приложение В
ИСХОДНЫЙ КОД КЛИЕНТСКОЙ ЧАСТИ
(рекомендуемое)

Листинг В.1 – AuthorizationVC.swift

```
//
//  AuthorizationVC.swift
//  wannaholiday
//
//  Created by MelnikovK on 03.11.2017.
//  Copyright © 2017 sofinvent. All rights reserved.
//

import UIKit
import SnapKit
import VK_ios_sdk
import GoogleSignIn
import KeychainAccess
import FBSDKCoreKit
import FacebookLogin
import AuthenticationServices

class AuthorizationVC: UIViewController, WHNavigationDelegate {

    enum Suberror: String {
        case authWithDisconnectedNetwork = "Авторизация через соцсет  
ь не удалась. Попробуйте войти в приложение по номеру телефона и  
привязать аккаунт выбранной соцсети"
    }

    let activityIndicatorView = BlockingActivityIndicatorView(frame: .zero)

    lazy var navigationBar: WHNavigationBar = {
        let navBar = WHNavigationBar(frame: CGRect(x: 0, y: 0, width: 375, height: WHNavigationBar.preferableHeight))
        navBar.navigationDelegate = self
        navBar.title = "Вход"
        navBar.color(color: .red)
        return navBar
    }()

    lazy var scrollView: UIScrollView = {
        let scroll = UIScrollView()
        scroll.isUserInteractionEnabled = true
        return scroll
    }()

    let containerView = UIView()

    let callToActionPhoneLabel: UILabel = {
        let label = UILabel()
        label.text = "Введите номер своего телефона"
        label.textColor = UIColor(red: 0, green: 0, blue: 0, alpha: 0.87)
    }()
```

```

    label.font = UIFont(name: "Roboto-Medium", size: 16)
    label.textAlignment = .center
    return label
}()

lazy var phoneTextField: UITextField = {
    let textField = UITextField()
    textField.keyboardType = .phonePad
    textField.backgroundColor = .clear
    textField.textAlignment = .center
    textField.font = UIFont(name: "Roboto-Medium", size: 18)
    textField.placeholder = "(000) 000 00 00"

    let plusSevenLabel = UILabel()
    plusSevenLabel.text = "+7"
    plusSevenLabel.font = UIFont(name: "Roboto-Medium", size: 18)
    plusSevenLabel.textAlignment = .center

    let container = UIView()
    container.snp.makeConstraints {
        $0.size.equalTo(CGSize(width: 30, height: 36))
    }
    container.addSubview(plusSevenLabel)
    plusSevenLabel.snp.makeConstraints {
        $0.left.equalToSuperview().offset(10)
        $0.centerY.equalToSuperview()
    }
    textField.leftView = container
    textField.leftViewMode = .always

    textField.delegate = self
    return textField
}()

let separator: UIView = {
    let view = UIView()
    view.backgroundColor = .lightGray
    return view
}()

lazy var instructionLabel: UILabel = {
    let label = UILabel()
    label.text = "После нажатия на кнопку вам поступит звонок. Отвечать на него не нужно.\nЗапомните последние 4 цифры номера."
    label.textColor = UIColor(red: 0, green: 0, blue: 0, alpha: 0.87)
    label.textAlignment = .center
    label.numberOfLines = 0
    label.lineBreakMode = .byWordWrapping
    label.font = UIFont(name: "Roboto-Medium", size: 14)
    return label
}()

```

```

lazy var instructionImage: UIImageView = {
    let image = UIImageView()
    image.image = #imageLiteral(resourceName: "authorizationHint")
}()
return image
}()

lazy var continueButton: UIButton = {
    let button = UIButton()
    button.setTitle("ПОЗВОЛИТЬ МНЕ", for: .normal)
    button.titleLabel?.font = UIFont(name: "Roboto-Medium", size: 14)
    button.layer.cornerRadius = 18
    button.backgroundColor = #colorLiteral(red: 0.721216023, green: 0.7291184068, blue: 0.7498105168, alpha: 1)
    button.setTitleColor(#colorLiteral(red: 0.5563387275, green: 0.5678691864, blue: 0.6135072112, alpha: 1), for: .normal)
    button.addTarget(self, action: #selector(continueTap) , for: .touchUpInside)
    button.isUserInteractionEnabled = false
    button.layer.shadowColor = UIColor(red: 0, green: 0, blue: 0, alpha: 0.25).cgColor
    button.layer.shadowOffset = CGSize(width: 0.0, height: 2.0)
    button.layer.shadowOpacity = 1.0
    button.layer.shadowRadius = 2
    button.layer.masksToBounds = false
    return button
}()

lazy var socialInstructionLabel: UILabel = {
    let label = UILabel()
    label.text = "Если вы уже регистрировались и привязали социальные сети"
    label.textColor = UIColor(red: 0, green: 0, blue: 0, alpha: 0.54)
    label.textAlignment = .center
    label.numberOfLines = 0
    label.lineBreakMode = .byWordWrapping
    label.font = UIFont(name: "Roboto-Medium", size: 14)
    return label
}()

lazy var networkStackView: UIStackView = {
    let stack = UIStackView(arrangedSubviews: [])
    stack.alignment = .center
    stack.distribution = .equalCentering
    stack.spacing = 50
    return stack
}()

lazy var vkButton: UIButton = {
    let button = UIButton()

```

```

        button.addTarget(self, action: #selector(vkTap) , for: .touchUpInside)
        button.setBackgroundImage(#imageLiteral(resourceName: "vkButton"), for: .normal)
        button.isUserInteractionEnabled = true
        return button
    }()

    lazy var fbButton: UIButton = {
        let button = UIButton()
        button.addTarget(self, action: #selector(fbTap) , for: .touchUpInside)
        button.setBackgroundImage(#imageLiteral(resourceName: "fbButton"), for: .normal)
        button.isUserInteractionEnabled = true
        return button
    }()

    lazy var googleButton: UIButton = {
        let button = UIButton()
        button.addTarget(self, action: #selector(googleTap) , for: .touchUpInside)
        button.setBackgroundImage(#imageLiteral(resourceName: "googleIcon"), for: .normal)
        button.isUserInteractionEnabled = true
        return button
    }()

    lazy var appleButton: UIButton = {
        let button = UIButton()
        button.addTarget(self, action: #selector(appleSignInTapped), for: .touchUpInside)
        button.setBackgroundImage(#imageLiteral(resourceName: "appleButton"), for: .normal)
        button.isUserInteractionEnabled = true
        return button
    }()

    // MARK: - Properties

    private let countryCode = "7"

    // MARK: - Navigation bar button actions

    func leftButtonClicked() {
        backTap()
    }

    // MARK: - Button actions
    @objc func backTap() {
        UserDefaults.standard.set(false, forKey: "firstLaunch")
        navigationController?.popViewController(animated: true)
    }

```

```

@objc func continueTap() {
    proceedToConformSmsCode()
}

@objc func vkTap() {
    performAuthentication(with: .vk)
}

@objc func fbTap() {
    performAuthentication(with: .facebook)
}

@objc func googleTap() {
    performAuthentication(with: .google)
}

@objc func appleSignInTapped() {
    if #available(iOS 13.0, *) {
        let provider = ASAuthorizationAppleIDProvider()
        let request = provider.createRequest()
        request.requestedScopes = [.fullName, .email]

        let authController = ASAuthorizationController(authorizationRequests: [request])

        authController.presentationContextProvider = self
        authController.delegate = self
        authController.performRequests()
    }
}

private func proceedToConformSmsCode() {
    guard let phoneNumber = phoneTextField.text else { return }
    let numberWithCode = countryCode + phoneNumber
    phoneTextField.resignFirstResponder()
    activityIndicatorView.isHidden = false

    ServerManager.shared.triggerSmsCode(for: numberWithCode) { [
        unowned self] (isSuccessful, responseCode, response) in
        defer { DispatchQueue.main.async { self.activityIndicatorView.isHidden = true } }

        guard isSuccessful && responseCode! == 200 else {
            self.show(error: WHError(code: responseCode, response: response, handler: nil))
            return
        }

        guard let response = response as? [String : Bool], let userHasAccount = response["has_account"] else { return }

        if userHasAccount {

```

```

        DispatchQueue.main.async {
            self.navigationController?.pushViewController(AuthorizationAdmissionVC(phoneNumber: numberWithCode), animated: true)
        }
    } else {
        DispatchQueue.main.async {
            guard let registerVC = RegistrationVC.storyboardInstanceWith(phoneNumber: numberWithCode) else { return }
            self.navigationController?.pushViewController(registerVC, animated: true)
        }
    }
}

private func performAuthentication(with network: ServerManager.SocialType) {
    var provider: AuthProvider
    switch network {
    case .vk: provider = VKManager(with: self)
    case .facebook: provider = FacebookManager(with: self)
    case .google: provider = GoogleManager()
    case .apple: return
    }

    AuthorizationManager.shared.authorization(with: provider) {
        [weak self] token in
            guard let token = token else { return }

            var userId: String
            var authServerType: ServerManager.AuthType

            switch network {
            case .vk:
                userId = VKSdk.accessToken().userId
                authServerType = .vk
            case .facebook:
                userId = FBSDKAccessToken.current().userID
                authServerType = .facebook
            case .google:
                userId = GIDSignIn.sharedInstance().currentUser.userID
                authServerType = .google
            case .apple: return
            }

            self?.authenticate(withNetwork: authServerType, userId: userId, token: token, completion: { isSuccessful in
                guard isSuccessful else { return }
                self?.loadAccountData() {
                    self?.dismiss(animated: true, completion: nil)
                }
            })
    }
}

```

```

    }

    private func authenticate(withNetwork network: ServerManager.AuthType,
                              userId: String, token: String, completion: ((Bool) -
    > ())? = nil) {
        ServerManager.shared.auth(with: network, for: userId, with:
    token, with: { [weak self] (isSuccessful, responseCode, response
    ) in

            guard responseCode != nil else {
                self?.show(error: WLError(code: nil, suberrors: ["Ошибка
    авторизации"], handler: nil))
                return
            }

            guard isSuccessful else {
                self?.show(error: WLError(code: responseCode, suberrors:
    ((response as? Dictionary<String, Any>)?["errors"] as? String).
    flatMap{ [ $0 ]}, handler: nil))
                switch network {
                    case .vk:
                        VKSdk.forceLogout()
                    case .facebook:
                        LoginManager().logout()
                    case .google:
                        GIDSignIn.sharedInstance()?.signOut()
                    case .apple:
                        print("apple")
                    case .phone:
                        print("phone")
                }
                return
            }

            guard responseCode != 401 else {
                self?.show(error: WLError(code: responseCode, suberrors:
    [Suberror.authWithDisconnectedNetwork.rawValue], handler: nil))
                return
            }

            guard responseCode != 200 && isSuccessful else {
                self?.show(error: WLError(code: responseCode, response:
    response, handler: nil))
                return
            }

            guard let userToken = (response as? Dictionary<String, Any
    >)?["access_token"] as? String else {
                self?.show(error: WLError(code: responseCode, suberrors:
    ["Наш сервис не смог Вас авторизовать!"], handler: nil))
                return
            }

            do {

```

```

        try Keychain().store(data: userToken, for: .authorizationToken)
        completion?(true)
    } catch {
        self?.show(error: NSError(code: nil, suberrors: ["Сохранение авторизационных данных не удалось!"], handler: nil))
    }
    })
}

//get rid of it if everything works fine
private func loadAccountData(completion: (() -> ())? = nil) {
    AccountManager.shared.fetchAccountData { [weak self] result
in
        DispatchQueue.main.async {
            switch result {
                case .success(let account): AccountManager.shared.account = account; completion?()
                case .nonAuthorized: completion?()
                case .failed(let fetchError): self?.show(error: fetchError)
or)
            }
        }
    }
}

// MARK: - Lifecycle
override func viewDidLoad() {
    super.viewDidLoad()

    view.backgroundColor = .white

    view.addSubview(navigationBar)

    view.addSubview(containerView)

    containerView.addSubview(callToActionPhoneLabel)
    containerView.addSubview(phoneTextField)
    containerView.addSubview(separator)
    containerView.addSubview(instructionLabel)
    containerView.addSubview(instructionImage)
    containerView.addSubview(continueButton)
    //containerView.addSubview(socialInstructionLabel)
    containerView.addSubview(activityIndicatorView)

    navigationBar.snp.makeConstraints {
        $0.height.equalTo(54)
        $0.top.equalTo(view.snp.topMargin)
        $0.leading.equalToSuperview()
        $0.trailing.equalToSuperview()
    }

    // Scroll View

```

```

containerView.snp.makeConstraints {
    $0.top.equalTo(navigationBar.snp.bottom)
    $0.centerX.equalToSuperview()
    $0.width.equalToSuperview()
    $0.bottom.equalToSuperview()
}

// Target Views
callToActionPhoneLabel.snp.makeConstraints {
    $0.top.equalTo(containerView.snp.top).offset(34)
    $0.centerX.equalToSuperview()
    $0.size.equalTo(CGSize(width: 328, height: 18))
}

1) phoneTextField.snp.makeConstraints {
    $0.top.equalTo(callToActionPhoneLabel.snp.bottom).offset(2)
    $0.centerX.equalToSuperview()
    $0.size.equalTo(CGSize(width: 200, height: 43))
}

separator.snp.makeConstraints {
    $0.height.equalTo(1)
    $0.width.equalTo(phoneTextField)
    $0.centerX.equalTo(phoneTextField)
    $0.top.equalTo(phoneTextField.snp.bottom)
}

instructionLabel.snp.makeConstraints {
    $0.top.equalTo(separator.snp.bottom).offset(24)
    $0.centerX.equalToSuperview()
    $0.size.equalTo(CGSize(width: 328, height: 58))
}

instructionImage.snp.makeConstraints {
    $0.top.equalTo(instructionLabel.snp.bottom).offset(16)
    $0.centerX.equalToSuperview()
    $0.size.equalTo(CGSize(width: 220, height: 154))
}

continueButton.snp.makeConstraints {
    $0.top.equalTo(instructionImage.snp.bottom).offset(24)
    $0.centerX.equalToSuperview()
    $0.size.equalTo(CGSize(width: 150, height: 36))
}

activityIndicatorView.snp.makeConstraints {
    $0.size.equalToSuperview()
    $0.center.equalToSuperview()
}

GIDSignIn.sharedInstance().uiDelegate = self
}

```

```

private func setupSocialButtons() {
    containerView.addSubview(networkStackView)
    networkStackView.snp.makeConstraints {
        $0.top.equalTo(socialInstructionLabel.snp.bottom).offset(1
6)
        $0.centerX.equalToSuperview()
        $0.height.equalTo(40)
        $0.bottom.equalTo(containerView.snp.bottom).inset(20)
    }

    if #available(iOS 13, *) {
        networkStackView.addArrangedSubview(appleButton)
        appleButton.snp.makeConstraints {
            $0.size.equalTo(CGSize(width: 40, height: 40))
        }
    }
    networkStackView.addArrangedSubview(googleButton)
    networkStackView.addArrangedSubview(vkButton)
    networkStackView.addArrangedSubview(fbButton)

    vkButton.snp.makeConstraints {
        $0.size.equalTo(CGSize(width: 40, height: 40))
    }

    fbButton.snp.makeConstraints {
        $0.size.equalTo(CGSize(width: 40, height: 40))
    }

    googleButton.snp.makeConstraints {
        $0.size.equalTo(CGSize(width: 40, height: 40))
    }
}

override func viewWillAppear(_ animated: Bool) {
    super.viewWillAppear(animated);
    UIApplication.shared.statusBarView?.backgroundColor = #color
Literal(red: 0.5725490196, green: 0.01568627451, blue: 0.0431372
549, alpha: 1)
    self.navigationController?.isNavigationBarHidden = true
}

// MARK: - Networking Delegates
extension AuthorizationVC: VKSdkUIDelegate {
    func vkSdkShouldPresent(_ controller: UIViewController!) {
        print("VK app not found. loaded webView authorization")
        present(controller, animated: true)
    }

    func vkSdkNeedCaptchaEnter(_ captchaError: VKError!) {
        print("Ошибка")
        print(captchaError)
    }
}

```

```

    }
}

extension AuthorizationVC: GIDSignInUIDelegate { }
extension AuthorizationVC: ErrorThrowingVC {}

extension AuthorizationVC: UITextFieldDelegate {

    func textField(_ textField: UITextField, shouldChangeCharacter
sIn range: NSRange, replacementString string: String) -> Bool {
        guard textField == phoneTextField else { return true }

        var numberWithCode = countryCode + phoneTextField.text!
        if !string.isEmpty {
            numberWithCode += string
        } else {
            numberWithCode.remove(at: numberWithCode.index(before: num
berWithCode.endIndex))
        }

        switch Validator.shared.phoneNumberValidated(numberWithCode)
        {
        case .failure:
            continueButton.backgroundColor = #colorLiteral(red: 0.7212
16023, green: 0.7291184068, blue: 0.7498105168, alpha: 1)
            continueButton.setTitleColor(#colorLiteral(red: 0.55633872
75, green: 0.5678691864, blue: 0.6135072112, alpha: 1), for: .no
rmal)
            continueButton.isUserInteractionEnabled = false
        case .success:
            continueButton.backgroundColor = #colorLiteral(red: 0.9921
568627, green: 0.1215686275, blue: 0.1803921569, alpha: 1)
            continueButton.setTitleColor(#colorLiteral(red: 1, green:
1, blue: 1, alpha: 1), for: .normal)
            continueButton.isUserInteractionEnabled = true
        }
        return true
    }
}

extension AuthorizationVC: ASAuthorizationControllerPresentation
ContextProviding {

    @available(iOS 13.0, *)
    func presentationAnchor(for controller: ASAuthorizationControl
ler) -> ASPresentationAnchor {
        return self.view.window!
    }
}

```

```

extension AuthorizationVC: ASAuthorizationControllerDelegate {

    @available(iOS 13.0, *)
    func authorizationController(controller: ASAuthorizationContro
ller, didCompleteWithError error: Error) {
        print("authorization error")
        guard let error = error as? ASAuthorizationError else { retu
rn }
        //
    }

    @available(iOS 13.0, *)
    func authorizationController(controller: ASAuthorizationContro
ller, didCompleteWithAuthorization authorization: ASAuthorizatio
n) {

        if let appleIDCredential = authorization.credential as? ASAu
thorizationAppleIDCredential {
            guard let data = appleIDCredential.authorizationCode else
{ return }

            let token = String(decoding: data, as: UTF8.self)
            let userId = appleIDCredential.user

            authenticate(withNetwork: .apple, userId: userId, token: t
oken, completion: { [weak self] isSuccessful in
                guard isSuccessful else { return }
                self?.loadAccountData() {
                    self?.dismiss(animated: true, completion: nil)
                }
            })
        }
    }
}

```

Листинг В.2 – CompilationReviewsVC.swift

```

//
//  CompilationReviewsVC.swift
//  wannaholiday
//
//  Created by Егор Ермин on 21.09.2020.
//  Copyright © 2020 sofinvent. All rights reserved.
//

import UIKit

class CompilationReviewsVC: UIViewController {

    @IBOutlet weak var navigationBar: WHNavigationBar!
    @IBOutlet weak var reviewsTableView: UITableView!
    @IBOutlet weak var firstReviewHint: UIView!

```

```

@IBOutlet weak var purchaseHintLabel: UILabel!
@IBOutlet weak var commentTextView: UITextView! {
    didSet {
        setupPlaceholderTextView()
    }
}
@IBOutlet weak var commentViewButton: UIButton!
@IBOutlet weak var commentGroupView: UIView!
@IBOutlet weak var deleteView: DeleteReviewDialog!
@IBOutlet weak var sendCommentButton: UIButton! {
    didSet {
        let image = UIImage(named: "sendIcon")?.withRenderingMode(
.alwaysTemplate)
        sendCommentButton.setImage(image, for: .normal)
    }
}
@IBOutlet weak var deleteCommentButton: UIButton! {
    didSet {
        let image = UIImage(named: "deleteIcon")?.withRenderingMod
e(.alwaysTemplate)
        deleteCommentButton.setImage(image, for: .normal)
    }
}
@IBOutlet weak var userCommentView: UIView! {
    didSet {
        userCommentView.layer.shadowColor = #colorLiteral(red: 0,
green: 0, blue: 0, alpha: 0.25)
        userCommentView.layer.shadowOpacity = 1
        userCommentView.layer.shadowRadius = 8
        userCommentView.layer.shadowOffset = CGSize(width: 0, heig
ht: -2)
    }
}

// Отступ таблицы с отзывами от нижнего края экрана
@IBOutlet weak var bottomTableViewConstraint: NSLayoutConstraint!

// Id подборки
private var compilationId: Int?
// Приобретена ли покупка
private var isPurchased: Bool = false
// Состояние видимости панели собственного отзыва
private var commentViewIsVisible: Bool = true
// Имеет ли пользователь собственный отзыв
private var isHaveUserComment: Bool = false {
    didSet {
        if isHaveUserComment {
            changeDeleteButtonState(isEnabled: true)
        } else {
            changeDeleteButtonState(isEnabled: false)
        }
    }
}

```

```

}
// Состояние индикатора прогресса
private var isActivateProgress: Bool = true
// Массив всех отзывов
private var allComments: [Comment?] = [] {
    didSet {
        if allComments.count == 0 {
            DispatchQueue.main.async {
                self.firstReviewHint.isHidden = false
            }
        } else {
            DispatchQueue.main.async {
                self.firstReviewHint.isHidden = true
            }
        }
    }
}
}
// Текст отзыва пользователя
private var commentText: String? = nil {
    didSet {
        if commentText != nil {
            self.changeSendButtonState(isEnabled: true)
        } else {
            self.changeSendButtonState(isEnabled: false)
        }
    }
}
}

private let activityIndicator = LoadingViewController()

//MARK: - Lifecycle
override func viewDidLoad() {
    super.viewDidLoad()
    setupNavigationBar()
    setupDeleteView()
    animateCommentView()
    fetchReviews(completion: { [weak self] result in
        if let review = try? result.get() {
            self?.allComments = self?.setAllComments(reviews: review
) ?? []
            DispatchQueue.main.async {
                self?.reviewsTableView.reloadData()
            }
        }
        DispatchQueue.main.async {
            self?.activityIndicator.remove()
        }
    })

    self.view.bringSubview(toFront: deleteView)
}

```

```

override func viewWillAppear(_ animated: Bool) {
    super.viewWillAppear(true)
    if !isPurchased || !AccountManager.shared.isUserAuthorized {
        DispatchQueue.main.async {
            self.commentGroupView.isHidden = true
            self.purchaseHintLabel.isHidden = false
        }
    }
}

override func viewDidAppear(_ animated: Bool) {
    super.viewDidAppear(animated)
    // Observer за скрывтием клавиатуры. Из за бага IQkeyboardManager приходится обновлять view
    NotificationCenter.default.addObserver(self, selector: #selector(reloadView), name: NSNotification.Name.UIKeyboardWillHide, object: nil)
}

override func viewDidDisappear(_ animated: Bool) {
    super.viewDidDisappear(animated)
    NotificationCenter.default.removeObserver(self)
}

// MARK: - Setup UI
private func setupNavigationBar() {
    navigationBar.navigationDelegate = self
    navigationBar.color(color: .red)
    navigationBar.leftNavItem = .close
    navigationBar.title = "ОТЗЫВЫ"
    navigationBar.setShadow(with: CGSize(width: 0, height: 4), radius: 3, color: #colorLiteral(red: 0, green: 0, blue: 0, alpha: 0.25))
    navigationBar.layer.shadowOpacity = 1
    navigationBar.layer.shadowRadius = 4

    self.view.bringSubview(toFront: navigationBar)
}

private func setupDeleteView() {
    deleteView.actionHandler = {
        self.processDeletingReview()
        self.deleteView.isHidden = true
    }
}

private func setupPlaceholderTextView() {
    DispatchQueue.main.async {
        self.commentTextView.text = "Как вам конкурсы из подборки?"
        self.commentTextView.textColor = #colorLiteral(red: 0.555, green: 0.569, blue: 0.61, alpha: 1)
    }
}

```

```

    }

    private func changeSendButtonState(isEnabled: Bool) {
        if isEnabled {
            DispatchQueue.main.async {
                self.sendCommentButton.isUserInteractionEnabled = true
                self.sendCommentButton.tintColor = #colorLiteral(red: 0.9921568627, green: 0.1215686275, blue: 0.1803921569, alpha: 1)
            }
        } else {
            DispatchQueue.main.async {
                self.sendCommentButton.isUserInteractionEnabled = false
                self.sendCommentButton.tintColor = #colorLiteral(red: 0.8549019608, green: 0.8549019608, blue: 0.8549019608, alpha: 1)
            }
        }
    }

    private func changeDeleteButtonState(isEnabled: Bool) {
        if isEnabled {
            DispatchQueue.main.async {
                self.deleteCommentButton.isUserInteractionEnabled = true
                self.deleteCommentButton.tintColor = #colorLiteral(red: 0.9921568627, green: 0.1215686275, blue: 0.1803921569, alpha: 1)
            }
        } else {
            DispatchQueue.main.async {
                self.deleteCommentButton.isUserInteractionEnabled = false
                self.deleteCommentButton.tintColor = #colorLiteral(red: 0.8549019608, green: 0.8549019608, blue: 0.8549019608, alpha: 1)
            }
        }
    }

    // MARK: - Animations
    // Анимация открытия/закрытия панели отзыва пользователя
    private func animateCommentView() {
        UIView.animate(withDuration: 0.2, animations: {
            if self.commentViewIsVisible {
                self.userCommentView.transform = CGAffineTransform(translationX: 0, y: 122)
                self.commentViewButton.setImage(UIImage(named: "upArrowIcon"), for: .normal)
                self.commentViewIsVisible = false
                self.bottomTableViewConstraint.constant = 40
            } else {
                self.userCommentView.transform = CGAffineTransform(translationX: 0, y: 0)
                self.commentViewButton.setImage(UIImage(named: "downArrowIcon"), for: .normal)
                self.commentViewIsVisible = true
            }
        })
    }

```

```

        DispatchQueue.main.asyncAfter(deadline: .now() + 0.2, execute: {
            self.bottomTableViewConstraint.constant = 162
        })
    }
})
}

// Анимация вращения. Рекурсивная функция
private func rotationAnimation(_ view: UIView) {
    if isActivateProgress {
        UIView.animate(withDuration: 0.5, delay: 0, options: [.curveLinear], animations: {
            view.transform = view.transform.rotated(by: .pi)
            UIView.animate(withDuration: 0.5, animations: {
                view.transform = view.transform.rotated(by: .pi)
            })
        }, completion: { finished in
            self.rotationAnimation(view)
        })
    }
}

// Запрос отзывов с сервера
private func fetchReviews(completion: @escaping (Result<Review, Error>) -> Void) {
    guard let id = compilationId else { return }
    DispatchQueue.main.async {
        self.add(self.activityIndicator)
    }
    ServerManager.shared.compilationsReviews(compilationId: id,
    completionHandler: { (_, _, response) in
        guard let jsonString = response as? String else { return }
        if let review = try? JSONDecoder().decode(Review.self, from: Data(jsonString.utf8)) {
            completion(.success(review))
        } else {
            completion(.failure(NetworkError()))
        }
    })
}

// Удаление отзыва
private func deleteComment(_ completion: ((_ isDeleted: Bool) -> ()))? {
    guard let id = compilationId else { return }

    ServerManager.shared.deleteCompilationReview(for: id, completionHandler: { (_, responseCode, response) in
        guard responseCode == 200 else {
            completion?(false)
            return
        }
    })
}

```

```

        completion?(true)
    })
}

private func processDeletingReview() {
    isActivateProgress = true
    // Замена иконки удаления отзыва на иконку индикатора прогресса
    deleteCommentButton.setImage(UIImage(named: "progressIcon"),
    for: .normal)
    // Начало анимации вращения
    rotationAnimation(deleteCommentButton)

    deleteComment() { [weak self] isDeleted in
        if isDeleted {
            self?.isHaveUserComment = false
            self?.allComments.remove(at: 0)
            DispatchQueue.main.async {
                self?.reviewsTableView.reloadData()
            }
        }
        let image = UIImage(named: "deleteIcon")?.withRenderingMode(.alwaysTemplate)
        self?.isActivateProgress = false
        DispatchQueue.main.async {
            self?.commentText = nil
            self?.setupPlaceholderTextView()
            self?.deleteCommentButton.layer.removeAllAnimations()
            self?.deleteCommentButton.setImage(image, for: .normal)
            self?.animateCommentView()
        }
    }
}

// Отправка отзыва на сервер
private func sendComment(completion: @escaping (Result<Comment, Error>) -> Void) {
    guard let commentText = commentText,
          !commentText.isEmpty,
          let id = compilationId else { return }
    let action: ServerManager.CommentAction = isHaveUserComment ? .update : .create

    ServerManager.shared.postCompilationsReviews(action, compilationId: id, review: commentText, responseHandler: { (_, responseCode, response) in
        guard responseCode == 200, let jsonString = response as? String else {
            completion(.failure(NetworkError()))
            return
        }
        if let userComment = try? JSONDecoder().decode(Comment.self, from: Data(jsonString.utf8)) {

```

```

        completion(.success(userComment))
    } else {
        completion(.failure(NetworkError()))
    }
})
}

private func processSendingReview() {
    isActivateProgress = true
    // Замена иконки отправки отзыва на иконку индикатора прогресса
    sendCommentButton.setImage(UIImage(named: "progressIcon"), for: .normal)
    // Начало анимации вращения
    rotationAnimation(sendCommentButton)
    sendComment() { [weak self] result in
        if let comment = try? result.get(), let isHave = self?.isHaveUserComment {
            // Если пользователь имел отзыв, заменяем его на обновленный
            if isHave {
                self?.allComments[0] = comment
                // Вставка отзыва пользователя в начало
            } else {
                self?.allComments.insert(comment, at: 0)
            }
            self?.isHaveUserComment = true
            DispatchQueue.main.async {
                self?.reviewsTableView.reloadData()
            }
        }
        let image = UIImage(named: "sendIcon")?.withRenderingMode(.alwaysTemplate)
        self?.isActivateProgress = false
        DispatchQueue.main.async {
            self?.sendCommentButton.layer.removeAllAnimations()
            self?.sendCommentButton.setImage(image, for: .normal)
            self?.animateCommentView()
            self?.commentTextView.resignFirstResponder()
        }
    }
}

private func setAllComments(reviews: Review) -> [Comment?] {
    var comments = reviews.reviews
    if let myReview = reviews.myReview {
        DispatchQueue.main.async {
            self.commentTextView.textColor = .black
            self.commentTextView.text = myReview.content
            self.changeSendButtonState(isEnabled: true)
        }
        comments.insert(myReview, at: 0)
        isHaveUserComment = true
    }
}

```

```

    } else {
        changeSendButtonState(isEnabled: false)
    }
    return comments
}

//MARK: - IBActions
@IBAction func moveCommentView(_ sender: Any) {
    animateCommentView()
}

@IBAction func publishAction(_ sender: Any) {
    processSendingReview()
}

@IBAction func deleteAction(_ sender: Any) {
    deleteView.isHidden = false
}

@IBAction func swipeUpHandler(_ gestureRecognizer: UISwipeGestureRecognizer) {
    if gestureRecognizer.state == .ended && !commentViewIsVisible {
        animateCommentView()
    }
}

@IBAction func swipeDownHandler(_ gestureRecognizer: UISwipeGestureRecognizer) {
    if gestureRecognizer.state == .ended && commentViewIsVisible {
        animateCommentView()
    }
}

/* Обновляем view после скрывтия клавиатуры.
   Иначе из-за бага IQKeyboard, некорректно отображается Navigation Bar */
@objc func reloadView() {
    self.view.setNeedsLayout()
    self.view.layoutIfNeeded()
}

//MARK: - UITableViewDelegate, UITableViewDataSource
extension CompilationReviewsVC: UITableViewDelegate, UITableViewDataSource {
    func tableView(_ tableView: UITableView, numberOfRowsInSectionSection: Int) -> Int {
        return allComments.count
    }
}

```

```

    func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
        let cell = tableView.dequeueReusableCell(withIdentifier: "CompilationReviewCell", for: indexPath) as! CompilationReviewTableViewCell
        guard let comment = allComments[indexPath.row] else { return cell }
        cell.configureCell(comment: comment)
        return cell
    }
}

// MARK: - UITextViewDelegate
extension CompilationReviewsVC: UITextViewDelegate {
    func textViewDidChange(_ textView: UITextView) {
        if textView.text != "", !textView.text.trimmingCharacters(in: .whitespaces).isEmpty {
            self.commentText = textView.text
        } else {
            self.commentText = nil
        }
    }

    func textViewDidBeginEditing(_ textView: UITextView) {
        if textView.textColor == #colorLiteral(red: 0.555, green: 0.569, blue: 0.61, alpha: 1) {
            textView.text = nil
            textView.textColor = UIColor.black
        }
    }

    func textViewDidEndEditing(_ textView: UITextView) {
        if textView.text.isEmpty {
            setupPlaceholderTextView()
        }
    }
}

//MARK: - WHNavigationDelegate
extension CompilationReviewsVC: WHNavigationDelegate {
    func leftButtonClicked() {
        DispatchQueue.main.async {
            self.dismiss(animated: true, completion: nil)
        }
    }
}

extension CompilationReviewsVC {
    static func storyboardInstance(_ compilationId: Int, _ purchaseStatus: Bool) -> CompilationReviewsVC? {

```

```

        let stb = UIStoryboard(name: String(describing: "Compilation
Reviews"), bundle: nil)
        let vc = stb.instantiateInitialViewController() as? Compilat
ionReviewsVC
        vc?.compilationId = compilationId
        vc?.isPurchased = purchaseStatus
        return vc
    }
}

```

Листинг В.3 – ServerManager.swift

```

//
//  ServerManager.swift
//  wanna-holiday
//
//  Created by iOS Developer 1 on 14.12.16.
//  Copyright © 2016 sofinvent. All rights reserved.
//

import Just
import ReachabilitySwift
import FirebaseInstanceID

class ServerManager {

    enum AuthType: String { case phone, vk, facebook, google,
apple }
    enum CommentAction { case create, update }
    enum ResponseType { case text, json }
    enum RequestType { case get, post, put, delete, postData,
patch }
    enum SocialType: String { case vk, facebook, google, apple }

    enum URI: String {
        case account, accountAuth = "account/auth_method",
        accountVK = "account/auth_method/vk", accountFB =
"account/auth_method/facebook", accountGoogle =
"account/auth_method/google",
        adverts, agreement, applicationData = "events", auth,
        cities, count = "search/user_services/count",
        favorite, favorites, fastSearchUserService =
"fast_search_user_service", feedback, filters = "filter",
        gameHandsUp = "games/handsup/cards",
        hot = "hot",
        like, logout,
        mytenders, myClosedTenders = "mytenders/closed",
mysimplefeasts,
        search = "search/user_services",
        register, report, reviews, refreshFcmToken =
"refresh_fcm_token",

```

```

    statistics = "stat", statCall = "stat/call", callAgent =
"stat/call_agent", statVisit = "stat/visit", statConstructor =
"stat/constructor", statUserservice = "stat/userservice",
    triggerSmsCode = "sms_token",
    tendersAll = "tenders/all",
    tendersBids = "tenders/bids",
    tenders = "tenders",
    tendersContracts = "tenders/contracts",
    serviceProviderSummary = "service_provider/summary",
    userService = "user_service",
    serviceProviderForm = "service_provider/service_form",
serviceFormDistricts =
"service_provider/service_form/districts",
    services = "service_provider/services", serviceProviderFiles
= "service_provider/files",
    purchase = "purchase", serviceEnhancements =
"purchase/service_enhancements", serviceEnhancement =
"purchase/service_enhancement",
    contestCompilationEnhancement =
"purchase/contest_compilation", contestCompilations =
"contest_compilations"
}

```

```

enum ParameterType: String {
    case action, authCredentials = "token", authIdentifier =
"id", authType = "provider", auth = "auth", appVersion =
"app_version",
    birthDate = "birthdate",
    cause, cityId = "city_id", comment = "content", rating,
context = "context",
    email, eventId = "event_id",
    filterId = "role_filter_id", fcmToken = "fcm_token", fee,
hometown = "hometown",
    imageFile = "image[file]", isFavorite = "is_favorite",
isLiked = "is_liked", instanceId = "instance_id",
    lastname, limit,
    message,
    name = "name",
    offset,
    page, password, phoneNumber = "phone",
    os,
    referrer = "referrer", referrerType = "referrer_type",
    serviceId = "service_id", smsCode = "code", sortingId =
"rand", subject = "subject",
    timeStamp = "ts",
    userServiceId = "user_service_id", userType = "type",
    version = "last_time_updated", viaSocial = "via",
    // ServiceProvider
    serviceProviderID = "service_type_id",
    //Tenders Parameter Type
    beginTime = "begin_time",
    comments = "comments",
    endTime = "end_time",

```

```

    maxCost = "max_cost", minCost = "min_cost",
    price = "price",
    status = "status",
    when = "when",
    //Game
    packId = "pack_id",
    question = "question", query = "query",
    author = "author",
    sku, purchaseToken = "purchase_token"
}

enum Status: String {
    case
    accepted = "accepted",
    applied = "applied",
    pending = "pending",
    rejected = "rejected"
}

enum ContextUserService: String {
    case
    constructor = "constructor",
    tenders = "tenders",
    hotoffers = "hotoffers",
    favorites = "favorites",
    search = "search",
    myfeasts = "myfeasts",
    push = "push",
    dynlink = "dynlink"
}

enum Header: String {
    case
    authorization = "Authorization",
    hash = "X-APP-HASH",
    lastModify = "If-Modified-Since",
    version = "X-APP-VERSION"
}

let defaultTimeout: Double = 60.0

private let apiVersion: (headerName: String, version: String)
= (headerName: Header.version.rawValue, version: "3")
private let reachability = Reachability()!
private let secretKey = "934567df6g8b70d9f7b87df78df7b"

let baseServerAddress: String = {
    guard let path = Bundle.main.path(forResource: "Info",
ofType: "plist"),

```

```

        let infoDict = NSDictionary(contentsOfFile: path) as?
[String: AnyObject],
        let urlDict = infoDict["Endpoint"],
        let url = urlDict["url"] as? String
        else { return ""}
        return url
    }()
// let baseServerAddress = "http://devholiday.osinit.com"
private(set) lazy var serverApiAddress: String = { return
baseServerAddress + "/api/" }()
private(set) lazy var serverLKAddress: String = { return
baseServerAddress + "/lk/login/mobile/" }()

// MARK: - Singleton
static let shared: ServerManager = { return ServerManager()
}()

private init() {}

private func request(on type: RequestType,
                    for uri: URI,
                    restfulId: Int? = nil,
                    params: [String: Any] = [:],
                    json: Any? = nil,
                    headers: [String: String] = [:],
                    files: [String: HTTPFile] = [:],
                    responseType: ResponseType = .json,
                    responseHandler: @escaping (Bool, Int?,
Any?) -> ()) -> () {

    func buildUrl(for resource: URI, with restfulId: Int?) ->
String {
        return serverApiAddress + resource.rawValue + (restfulId
!= nil ? "/" + String(restfulId!) : "")
    }

    let requestUrl = buildUrl(for: uri, with: restfulId)
    let timeStamp = Int(Date().timeIntervalSince1970)
    let timeStampedParams = params.conjoined(with:
(ParameterType.timeStamp.rawValue, timeStamp))
    #if DEBUG
    print("params --> \(params)")
    #endif
    let requestSignSHA256: String
    if !params.isEmpty, !files.isEmpty {
        requestSignSHA256 = hashingByHMACSHA256(params:
timeStampedParams, sorted: false)
    } else {
        requestSignSHA256 = hashingByHMACSHA256(params:
timeStampedParams, sorted: true)
    }
}

```

```

    let requestParams = timeStampedParams

    let requestHeaders = headers.conjoined(with:
apiVersion).conjoined(with: (Header.hash.rawValue,
requestSignSHA256))

    let resultHandler: ((HTTPResult) -> Void) = { response in
        let r = responseType == .text ? response.text :
response.json

        self.showLogs(response, ofType: responseType)
        if response.ok {
            guard let responseStatusCode = response.statusCode else
{ return }
            let isSuccess = ([200, 204, 304] as
Set<Int>).contains(responseStatusCode) // #
            responseHandler(isSuccess, responseStatusCode, r)
        } else {
            let responseCode = self.hasInternetConnection ?
response.statusCode : -200
            responseHandler(false, responseCode, r ?? nil)
        }
    }
    #if DEBUG
    print("requestParams --> \(requestParams)")
    #endif

    switch type {
    case .get: Just.get      (requestUrl, params:
requestParams, headers: requestHeaders, timeout: defaultTimeout)
{ resultHandler($0) }
    case .post: Just.post    (requestUrl, params:
requestParams, json: json, headers: requestHeaders, files:
files, timeout: defaultTimeout) { resultHandler($0) }
    case .put: Just.put      (requestUrl, params:
requestParams, json: json, headers: requestHeaders, timeout:
defaultTimeout) { resultHandler($0) }
    case .delete: Just.delete (requestUrl, params:
requestParams, headers: requestHeaders, timeout: defaultTimeout)
{ resultHandler($0) }
    case .postData: Just.post (requestUrl, data: requestParams,
headers: requestHeaders, files: files, timeout: defaultTimeout)
{ resultHandler($0) }
    case .patch: Just.patch  (requestUrl, data: requestParams,
headers: requestHeaders, files: files, timeout: defaultTimeout)
{ resultHandler($0) }
    }
}

private func requestURI(on type: RequestType,
                        for uri: String,
                        restfulId: Int? = nil,
                        params: [String: Any] = [:],

```

```

        headers: [String: String] = [:],
        files: [String: HTTPFile] = [:],
        responseType: ResponseType = .json,
        responseHandler: @escaping (Bool, Int?,
Any?) -> () -> () {

    func buildUrl(for resource: String, with restfulId: Int?) ->
String {
        return serverApiAddress + resource + (restfulId != nil ?
"/" + String(restfulId!) : "")
    }

    let requestUrl = buildUrl(for: uri, with: restfulId)

    let timeStamp = Int(Date().timeIntervalSince1970)
    let timeStampedParams = params.conjoined(with:
(ParameterType.timeStamp.rawValue, timeStamp))

    let requestSignSHA256 = hashingByHMACSHA256(params:
timeStampedParams, sorted: true)

    let requestParams = timeStampedParams

    let requestHeaders = headers.conjoined(with:
apiVersion).conjoined(with: (Header.hash.rawValue,
requestSignSHA256))

    let resultHandler: ((HTTPResult) -> Void) = { response in
        let r = responseType == .text ? response.text :
response.json

        self.showLogs(response, ofType: responseType)
        if response.ok {
            guard let responseStatusCode = response.statusCode else
{ return }
            let isSuccess = ([200, 204, 304] as
Set<Int>).contains(responseStatusCode) // #
            responseHandler(isSuccess, responseStatusCode, r)
        } else {
            let responseCode = self.hasInternetConnection ?
response.statusCode : -200
            responseHandler(false, responseCode, r ?? nil)
        }
    }

    switch type {
    case .get: Just.get      (requestUrl, params:
requestParams, headers: requestHeaders, timeout: defaultTimeout)
{ resultHandler($0) }
    case .post: Just.post    (requestUrl, params:
requestParams, headers: requestHeaders, files: files, timeout:
defaultTimeout) { resultHandler($0) }

```

```

        case .put: Just.put      (requestUrl, data:
requestParams, headers: requestHeaders, timeout: defaultTimeout)
{ resultHandler($0) }
        case .delete: Just.delete (requestUrl, params:
requestParams, headers: requestHeaders, timeout: defaultTimeout)
{ resultHandler($0) }
        case .postData: Just.post      (requestUrl, data:
requestParams, headers: requestHeaders, files: files, timeout:
defaultTimeout) { resultHandler($0) }
        case .patch: Just.patch(requestUrl, data: requestParams,
headers: requestHeaders, files: files, timeout: defaultTimeout)
{ resultHandler($0) }
    }
}

private func hashingByHMACSHA256(params: [String: Any],
sorted: Bool) -> String {
    if sorted {
        return Array(params.keys)
            .sorted(by: <)
            .compactMap { params[$0] }
            .map(String.init(describing:))
            .joined()
            .digest(.sha256, key: secretKey)
    } else {
        return Array(params.keys)
            .compactMap { params[$0] }
            .map(String.init(describing:))
            .joined()
            .digest(.sha256, key: secretKey)
    }
}

// MARK: - PublicAPI
var hasInternetConnection: Bool { return
reachability.isReachable }

func agreement(with responseHandler: @escaping (Bool, Int?,
Any?) -> ()) {
    request(on: .get, for: .agreement, responseType: .text,
responseHandler: responseHandler)
}

func applicationData(with lastModificationDate: String,
responseHandler: @escaping (Bool, Int?, Any?) -> ()) {
    request(on: .get, for: .applicationData, headers:
[Header.lastModify.rawValue: String(lastModificationDate)],
responseHandler: responseHandler)
}

func available(serviceId: Int, params: [String: Any]?,
responseHandler: @escaping (Bool, Int?, Any?) -> ()) {

```

```

        let cityId = params?[ParameterType.cityId.rawValue, default:
String(UserData.shared.location.cityId)] ??
String(UserData.shared.location.cityId)
        request(on: .get, for: .count, params:
[ParameterType.serviceId.rawValue: serviceId,
ParameterType.cityId.rawValue: cityId].merged(with:
params?.removingValue(for: ParameterType.cityId.rawValue)),
responseHandler: responseHandler)
    }

```

```

    func auth(with authType: AuthType, for authIdentifier: String,
with credentials: String, with responseHandler: @escaping (Bool,
Int?, Any?) -> ()) {
        guard let version =
Bundle.main.infoDictionary?["CFBundleShortVersionString"] as?
String else { return }
        InstanceID.instanceID().instanceID { (fcmToken, error) in
            var instanceId = ""
            if fcmToken?.token != nil { instanceId = fcmToken!.token }
            self.request(on: .get, for: .auth,
                params: [ParameterType.authType.rawValue:
authType.rawValue,
                        ParameterType.authIdentifier.rawValue:
authIdentifier,
                        ParameterType.authCredentials.rawValue:
credentials,
                        ParameterType.fcmToken.rawValue:
instanceId,
                        ParameterType.os.rawValue: "ios",
                        ParameterType.appVersion.rawValue:
version],
                responseHandler: responseHandler)
        }
    }

```

```

    }

    func cities(with lastModificationDate: String,
responseHandler: @escaping (Bool, Int?, Any?) -> ()) {
        request(on: .get, for: .cities, headers:
[Header.lastModify.rawValue: String(lastModificationDate)],
responseHandler: responseHandler)
    }

```

```

    func deleteReview(for userServiceId: Int, userToken: String,
responseHandler: @escaping (Bool, Int?, Any?) -> ()) {
        request(on: .delete, for: .reviews, restfulId:
userServiceId,
            headers: [Header.authorization.rawValue: userToken],
            responseHandler: responseHandler)
    }

```

```

    func filter(for eventId: Int, cityId: Int, serviceId: Int,
responseHandler: @escaping (Bool, Int?, Any?) -> ()) {
        request(on: .get, for: .filters, params:
[ParameterType.eventId.rawValue: eventId,
ParameterType.serviceId.rawValue: serviceId,
ParameterType.cityId.rawValue: cityId], responseHandler:
responseHandler)
    }

    func like(for userServiceId: Int, isPut: Bool, userToken:
String, responseHandler: @escaping (Bool, Int?, Any?) -> ()) {
        request(on: .postData, for: .like, // restfulId:
userServiceId,
            params: [ParameterType.userServiceId.rawValue:
userServiceId,
                ParameterType.isLiked.rawValue: isPut],
            headers: [Header.authorization.rawValue: userToken],
            responseHandler: responseHandler)
    }

    func loadFavorite(userToken: String, responseHandler:
@escaping (Bool, Int?, Any?) -> ()) {
        request(on: .get,
            for: .favorites,
            headers: [Header.authorization.rawValue: userToken],
            responseHandler: responseHandler)
    }

    func favorite(for userServiceId: Int, userToken: String,
responseHandler: @escaping (Bool, Int?, Any?) -> ()) {
        request(on: .postData,
            for: .favorites,
            params: [ParameterType.userServiceId.rawValue:
userServiceId],
            headers: [Header.authorization.rawValue: userToken],
            responseHandler: responseHandler)
    }

    func deleteFavorite(uri: String, userToken: String,
responseHandler: @escaping (Bool, Int?, Any?) -> ()) {
        requestURI(on: .delete,
            for: uri,
            headers: [Header.authorization.rawValue: userToken],
            responseHandler: responseHandler)
    }

    func news(for page: Int = 0, eventId: Int, cityId: Int,
responseHandler: @escaping (Bool, Int?, Any?) -> ()) {
        request(on: .get, for: .adverts,
            params: [ParameterType.page.rawValue: page,
                ParameterType.eventId.rawValue: eventId,
                ParameterType.cityId.rawValue: cityId],
            responseHandler: responseHandler)
    }

```

```

    }

    func post(feedback: String, from: String, with
responseHandler: @escaping (Bool, Int?, Any?) -> ()) {
        request(on: .post, for: .feedback, params:
[ParameterType.email.rawValue: from,
ParameterType.message.rawValue: feedback], responseHandler:
responseHandler)
    }

    func postReview(iWant action: CommentAction, review: String,
userServiceId: Int, userToken: String, responseHandler:
@escaping (Bool, Int?, Any?) -> ()) {
        let method: RequestType = action == .create ? .postData :
.put
        requestURI(on: method,
                    for: "reviews",
                    restfulId: userServiceId,
                    params: [ParameterType.comment.rawValue: review],
                    headers: [Header.authorization.rawValue: userToken],
                    responseHandler: responseHandler)
    }

    func post(statistics: StatisticsAction, for userServiceId:
Int, with responseHandler: @escaping (Bool, Int?, Any?) -> ()) {
        request(on: .post, for: .statistics, params:
[ParameterType.userServiceId.rawValue : userServiceId,
ParameterType.action.rawValue: statistics.rawValue],
responseHandler: responseHandler)
    }

    func postReportAboutCall(userServiceId: Int, instanceId:
String, auth: String, cityId: Int, with responseHandler:
@escaping (Bool, Int?, Any?) -> ()) {

        requestURI(on: .postData,
                    for: URI.statCall.rawValue,
                    params: [ParameterType.cityId.rawValue: cityId,
ParameterType.instanceId.rawValue:
instanceId,
ParameterType.auth.rawValue: auth,
ParameterType.userServiceId.rawValue :
userServiceId],
                    responseHandler: responseHandler)
    }

    //Sending a call report to the organizer on the server
    func reportAboutCallToAgent(instanceId: String, auth:
String, cityId: Int, with responseHandler: @escaping (Bool,
Int?, Any?) -> ()) {

        requestURI(on: .postData,
                    for: URI.callAgent.rawValue,

```

```

        params: [ParameterType.cityId.rawValue: cityId,
                  ParameterType.instanceId.rawValue: instanceId,
                  ParameterType.auth.rawValue: auth],
        responseHandler: responseHandler)
    }

    func statVisit(with responseHandler: @escaping (Bool, Int?,
Any?) -> ()) {
        let token = AccountManager.shared.authToken
        guard let cityId = RealmManager.city(for:
UserData.shared.location.cityId) else { return }

        InstanceID.instanceID().instanceID { (fcmToken, error) in
            var instanceId = ""
            if fcmToken?.token != nil { instanceId = fcmToken!.token }

            self.requestURI(on: .postData,
                           for: URI.statVisit.rawValue,
                           params: [ParameterType.cityId.rawValue:
cityId.id,
                                   ParameterType.instanceId.rawValue:
instanceId,
                                   ParameterType.auth.rawValue: token ??
""],
                           responseHandler: responseHandler)
        }
    }

    func statConstructor(instanceId: String, serviceId: Int,
cityId: Int, params: [String: Any]?, with responseHandler:
@escaping (Bool, Int?, Any?) -> ()) {

        requestURI(on: .postData,
                   for: URI.statConstructor.rawValue,
                   params: [ParameterType.cityId.rawValue: cityId,
                           ParameterType.instanceId.rawValue:
instanceId,
                           ParameterType.serviceId.rawValue:
serviceId].merged(with: params) ,
                   responseHandler: responseHandler)
    }

    func statUserService(userId: Int, context:
ContextUserService, params: [String: Any]?, with
responseHandler: @escaping (Bool, Int?, Any?) -> ()) {
        guard let cityId = RealmManager.city(for:
UserData.shared.location.cityId) else { return }

        InstanceID.instanceID().instanceID { (fcmToken, error) in
            var instanceId = ""
            if fcmToken?.token != nil { instanceId = fcmToken!.token }

            self.requestURI(on: .postData,

```

```

        for: URI.statUserservice.rawValue,
        params: [ParameterType.cityId.rawValue: cityId,
                  ParameterType.instanceId.rawValue:
instanceId,
                  ParameterType.userServiceId.rawValue :
userServiceId,
                  ParameterType.context.rawValue:
context.rawValue].merged(with: params),
        responseHandler: responseHandler)
    }
}

```

```

func reviews(forServiceId userServiceId: Int, offset: Int? =
nil, limit: Int? = nil, userToken: String? = nil,
responseHandler: @escaping (Bool, Int?, Any?) -> ()) {
    var params: [String: Int] = [:]
    if offset != nil { params = params.conjoined(with:
(ParameterType.offset.rawValue, offset!)) }
    if limit != nil { params = params.conjoined(with:
(ParameterType.limit.rawValue, limit!)) }
    let headers = userToken != nil ?
[Header.authorization.rawValue: userToken!] : [:]
    request(on: .get,
            for: .reviews,
            restfulId: userServiceId,
            params: params,
            headers: headers,
            responseType: .text,
            responseHandler: responseHandler)
}

```

```

func search(serviceId: Int, offset: Int, limit: UInt,
sortingId: UInt, params: [String: Any]?, responseHandler:
@escaping (Bool, Int?, Any?) -> ()) {
    let cityId = params?[ParameterType.cityId.rawValue] ??
String(UserData.shared.location.cityId)
    request(on: .get,
            for: .search,
            params: [ParameterType.serviceId.rawValue:
serviceId,
                    ParameterType.cityId.rawValue: cityId,
                    ParameterType.limit.rawValue: limit,
                    ParameterType.offset.rawValue: offset,
                    ParameterType.sortingId.rawValue:
sortingId].merged(
                    with: params?.removingValue(for:
ParameterType.cityId.rawValue)),
            responseHandler: responseHandler)
}

```

```

    func triggerSmsCode(for phoneNumber: String, with
responseHandler: @escaping (Bool, Int?, Any?) -> ()) {
        request(on: .postData, for: .triggerSmsCode, params:
[ParameterType.phoneNumber.rawValue : phoneNumber],
responseHandler: responseHandler)
    }

    func userService(for userServiceId: Int, userToken: String? =
nil, responseHandler: @escaping (Bool, Int?, Any?) -> ()) {
        let headers: [String: String] = (userToken != nil) ?
[Header.authorization.rawValue: userToken ?? ""] : [:]
        request(on: .get, for: .userService,
                params: [ParameterType.userServiceId.rawValue:
userServiceId],
                headers: headers,
                responseType: .text,
                responseHandler: responseHandler)
    }

    func fastSearchUserService(offset: Int, limit: UInt, subject:
String, params: [String: Any]?, responseHandler: @escaping
(Bool, Int?, Any?) -> ()) {
        let cityId = params?[ParameterType.cityId.rawValue] ??
String(UserData.shared.location.cityId)
        request(on: .get, for: .fastSearchUserService,
                params: [ParameterType.cityId.rawValue: cityId,
ParameterType.limit.rawValue: limit,
ParameterType.subject.rawValue: subject,
ParameterType.offset.rawValue:
offset].merged(
                    with: params?.removingValue(for:
ParameterType.cityId.rawValue)),
                responseHandler: responseHandler)
    }

    // Регистрация [POST /register]

    func register(for phoneNumber: String,
                  andSmsCode code: String,
                  name: String,
                  birthDate: String,
                  hometown: Int,
                  with responseHandler: @escaping (Bool,
Int?, Any?) -> ())
    {
        var optionalParams = [String: Any]()
        // проверяем есть ли instanceID
        var fcmToken: String? = nil
        InstanceID.instanceID().instanceID { (instanceId, error) in
            guard error == nil else { return }
            if let token = instanceId?.token {
                fcmToken = token
            }
        }
    }

```

```

    }
}
//если есть идентификатор, он передается в параметре
referrer
let refID = UserDefaults.standard.integer(forKey:
"referrerId")
if refID != 0 {
    optionalParams[ParameterType.referrer.rawValue] = refID
}
if fcmToken != nil {
    optionalParams[ParameterType.fcmToken.rawValue] =
fcmToken!
}
guard let version =
Bundle.main.infoDictionary?["CFBundleShortVersionString"] as?
String else { return }
request(on: .postData, for: .register,
        params: [ParameterType.phoneNumber.rawValue:
phoneNumber,
                 ParameterType.smsCode.rawValue: code,
                 ParameterType.name.rawValue: name,
                 ParameterType.birthDate.rawValue:
birthDate,
                 ParameterType.os.rawValue: "ios",
                 ParameterType.appVersion.rawValue: version,
                 ParameterType.hometown.rawValue:
hometown].merged(with: optionalParams),
        responseHandler: responseHandler)
}

//RefreshToken
func refreshFcmToken(refreshToken: String, token: String, with
responseHandler: @escaping (Bool, Int?, Any?) -> ()) {
    guard let version =
Bundle.main.infoDictionary?["CFBundleShortVersionString"] as?
String else { return }
    request(on: .postData,
            for: .refreshFcmToken,
            params: [ParameterType.fcmToken.rawValue:
refreshToken,
                   ParameterType.os.rawValue: "ios",
                   ParameterType.appVersion.rawValue:
version],
            headers: [Header.authorization.rawValue: token],
            responseHandler: responseHandler)
}

// Выход [POST /logout]
func logout(withAuthToken token: String, with responseHandler:
@escaping (Bool, Int?, Any?) -> ()) {
    request(on: .post, for: .logout, headers:
[Header.authorization.rawValue: token], responseHandler:
responseHandler)
}

```

```

    }

    // Профиль пользователя [/account?ts={ts}]
    func getAccount(withAuthToken token: String, with
responseHandler: @escaping (Bool, Int?, Any?) -> ()) {
        request(on: .get, for: .account, headers:
[Header.authorization.rawValue: token], responseType: .text,
responseHandler: responseHandler)
    }

    // Изменение профиля [POST]
    func update(name: String, birthDate: String = "", imageName:
String? = nil, imageData: Data? = nil, boundTo token: String,
with responseHandler: @escaping (Bool, Int?, Any?) -> ()) {
        var files: [String: HTTPFile] = [:]
        if let data = imageData, let imageName = imageName {
            let httpFile: HTTPFile = .data(imageName, data,
"image/jpeg")
            files = ["image[file]": httpFile]
        }

        request(on: .postData, for: .account,
                params: [ParameterType.name.rawValue: name,
                        ParameterType.birthDate.rawValue:
birthDate],
                headers: [Header.authorization.rawValue: token],
                files: files,
                responseType: .text, responseHandler:
responseHandler)
    }

    // Привязка аккаунтов соцсетей [POST
/account/auth_method?via={via}&id={id}&token={token}&ts={ts}]
    func connectSocial(withAuthToken token: String, socialType:
SocialType, socialId: String, socialToken: String, with
responseHandler: @escaping (Bool, Int?, Any?) -> ()) {
        request(on: .postData, for: .accountAuth,
                params: [ParameterType.viaSocial.rawValue:
socialType.rawValue,
                        ParameterType.authIdentifier.rawValue:
socialId,
                        ParameterType.authCredentials.rawValue:
socialToken],
                headers: [Header.authorization.rawValue: token],
                responseHandler: responseHandler)
    }

    // Отвязка аккаунтов соцсетей [DELETE
/account/auth_method/{provider}?ts={ts}]
    func disconnectSocial(withAuthToken token: String, socialType:
SocialType, with responseHandler: @escaping (Bool, Int?, Any?) -
> ()) {
        var uri: URI

```

```

switch socialType {
case .vk: uri = .accountVK
case .facebook: uri = .accountFB
case .google: uri = .accountGoogle
case .apple: return // need fix
}
request(on: .delete, for: uri,
        headers: [Header.authorization.rawValue: token],
        responseHandler: responseHandler)
}

//MARK: - Tenders

func loadMyTenders(withAuthToken token: String, with
responseHandler: @escaping (Bool, Int?, Any?) -> ()) {
    request(on: .get,
            for: .mytenders,
            headers: [Header.authorization.rawValue: token],
            responseHandler: responseHandler)
}

func loadMyClosedTenders(withAuthToken token: String, with
responseHandler: @escaping (Bool, Int?, Any?) -> ()) {
    request(on: .get,
            for: .myClosedTenders,
            headers: [Header.authorization.rawValue: token],
            responseHandler: responseHandler)
}

func loadUserInvitesTenders(withAuthToken token: String,
status: Status, with responseHandler: @escaping (Bool, Int?,
Any?) -> ()) {
    request(on: .get,
            for: .tenders,
            params: [ParameterType.status.rawValue: status],
            headers: [Header.authorization.rawValue: token],
            responseHandler: responseHandler)
}

func loadUserBidsTenders(withAuthToken token: String, with
responseHandler: @escaping (Bool, Int?, Any?) -> ()) {
    request(on: .get,
            for: .tendersBids,
            headers: [Header.authorization.rawValue: token],
            responseHandler: responseHandler)
}

func loadUserContractsTenders(withAuthToken token: String,
with responseHandler: @escaping (Bool, Int?, Any?) -> ()) {
    request(on: .get,
            for: .tendersContracts,
            headers: [Header.authorization.rawValue: token],
            responseHandler: responseHandler)
}

```

```

    func loadMyTenderWithId(withAuthToken token: String, tenderId:
Int, with responseHandler: @escaping (Bool, Int?, Any?) -> ()) {
        request(on: .get, for: .mytenders, restfulId: tenderId,
headers: [Header.authorization.rawValue: token],
            responseHandler: responseHandler)
    }

    func loadTenderWithUri(withAuthToken token: String, uri:
String, with responseHandler: @escaping (Bool, Int?, Any?) ->
()) {
        requestURI(on: .get,
                    for: uri,
                    headers: [Header.authorization.rawValue: token],
                    responseHandler: responseHandler)
    }

    func deleteBidFromTenderWithUri(withAuthToken token: String,
uri: String, with responseHandler: @escaping (Bool, Int?, Any?)
-> ()) {

        requestURI(on: .delete,
                    for: uri,
                    headers: [Header.authorization.rawValue: token],
                    responseHandler: responseHandler)

    }

    func deleteTenderWithUri(withAuthToken token: String, uri:
String, cause: String, with responseHandler: @escaping (Bool,
Int?, Any?) -> ()) {

        requestURI(on: .delete,
                    for: uri,
                    params: [ParameterType.cause.rawValue: cause],
                    headers: [Header.authorization.rawValue: token],
                    responseHandler: responseHandler)

    }

    func changeBidStatus(withAuthToken token: String, uri: String,
status: Status, with responseHandler: @escaping (Bool, Int?,
Any?) -> ()) {

        requestURI(on: .patch,
                    for: uri,
                    params: [ParameterType.status.rawValue: status],
                    headers: [Header.authorization.rawValue: token],
                    responseHandler: responseHandler)

    }

```

```

    func changeUserBidStatus(withAuthToken token: String, uri:
String, status: Status, price: Int?, fee: Int?, with
responseHandler: @escaping (Bool, Int?, Any?) -> ()) {
    var optionalParams = [String: Any]()
    if price != nil {
        optionalParams[ParameterType.price.rawValue] = price!
    }
    if fee != nil {
        optionalParams[ParameterType.fee.rawValue] = fee!
    }
    requestURI(on: .patch,
                for: uri,
                params: [ParameterType.status.rawValue:
status].merged(with: optionalParams),
                headers: [Header.authorization.rawValue: token],
                responseHandler: responseHandler)
}

//MARK: - User services

    func servicesList(withAuthToken token: String, query: String?,
offset: Int?, limit: Int?, with responseHandler: @escaping
(Bool, Int?, Any?) -> ()) {
    var optionalParams = [String: Any]()
    if query != nil {
        optionalParams[ParameterType.query.rawValue] = query!
    }
    if offset != nil {
        optionalParams[ParameterType.offset.rawValue] = offset!
    }
    if limit != nil {
        optionalParams[ParameterType.limit.rawValue] = limit!
    }
    request(on: .get,
            for: .services ,
            params: optionalParams,
            headers: [Header.authorization.rawValue: token],
            responseType: .text,
            responseHandler: responseHandler)
}

    func serviceProviderForm(for serviceId: Int, with
responseHandler: @escaping (Bool, Int?, Any?) -> ()) {
    guard let token = AccountManager.shared.authToken else {
return }

    request(on: .get,
            for: .serviceProviderForm,
            params: [ParameterType.serviceProviderID.rawValue:
serviceId],
            headers: [Header.authorization.rawValue: token],
            responseType: .text,
            responseHandler: responseHandler)
}

```

```

    }

    func serviceProviderForServiceId(serviceId: Int, with
responseHandler: @escaping (Bool, Int?, Any?) -> ()) {
        guard let token = AccountManager.shared.authToken else {
return }

        request(on: .get,
                for: .services,
                restfulId: serviceId,
                headers: [Header.authorization.rawValue: token],
                responseType: .text,
                responseHandler: responseHandler)
    }

    func deleteServiceProviderForServiceId(serviceId: Int, with
responseHandler: @escaping (Bool, Int?, Any?) -> ()) {
        guard let token = AccountManager.shared.authToken else {
return }

        request(on: .delete,
                for: .services,
                restfulId: serviceId,
                headers: [Header.authorization.rawValue: token],
                responseType: .text,
                responseHandler: responseHandler)
    }

    func createService(json: Any, with responseHandler: @escaping
(Bool, Int?, Any?) -> ()) {
        guard let token = AccountManager.shared.authToken else {
return }
        request(on: .post,
                for: .services,
                json: json,
                headers: [Header.authorization.rawValue: token],
                responseType: .text,
                responseHandler: responseHandler)
    }

    func updateService(serviceId: Int, json: Any, with
responseHandler: @escaping (Bool, Int?, Any?) -> ()) {
        guard let token = AccountManager.shared.authToken else {
return }
        request(on: .put,
                for: .services,
                restfulId: serviceId,
                json: json,
                headers: [Header.authorization.rawValue: token],
                responseType: .text,
                responseHandler: responseHandler)
    }

```

```

// Список районов предоставляемой услуги
func serviceDistrictForCityId(cityId: Int, with
responseHandler: @escaping (Bool, Int?, Any?) -> ()) {
    guard let token = AccountManager.shared.authToken else {
return }

    request(on: .get,
            for: .serviceFormDistricts,
            params: [ParameterType.cityId.rawValue: cityId],
            headers: [Header.authorization.rawValue: token],
            responseType: .text,
            responseHandler: responseHandler)
    }

// Отправка фото (создание услуги) [POST]
func serviceProviderFiles(imageName: String? = nil, imageData:
Data? = nil, token: String, with responseHandler: @escaping
(Bool, Int?, Any?) -> ()) {
    var files: [String: HTTPFile] = [:]
    if let data = imageData, let imageName = imageName {
        let httpFile: HTTPFile = .data(imageName, data,
"image/jpeg")
        files = ["image[file]": httpFile]
    }

    request(on: .post,
            for: .serviceProviderFiles,
            headers: [Header.authorization.rawValue: token],
            files: files,
            responseType: .text, responseHandler:
responseHandler)
    }

    func serviceProviderDeleteFile(fileId: Int, token: String,
with responseHandler: @escaping (Bool, Int?, Any?) -> ()) {
        request(on: .delete,
                for: .serviceProviderFiles,
                restfulId: fileId,
                headers: [Header.authorization.rawValue: token],
                responseHandler: responseHandler)
    }

// Запрос доступных опций
func getPaidOptions(with responseHandler: @escaping (Bool,
Int?, Any?) -> ()) {
    request(on: .get, for: .serviceEnhancements, params:
[ParameterType.os.rawValue: "ios"], responseType: .text,
responseHandler: responseHandler)
    }

// Запрос проверки токена покупки опции

```

```

    func validatePaidOption(sku: String, serviceId: Int, token:
String, with responseHandler: @escaping (Bool, Int?, Any?) ->
()) {
        guard let authToken = AccountManager.shared.authToken
else { return }
        request(on: .postData,
                for: .serviceEnhancement,
                params: [ParameterType.os.rawValue: "ios",
                        ParameterType.sku.rawValue: sku,
                        ParameterType.serviceId.rawValue:
serviceId,
                        ParameterType.purchaseToken.rawValue:
token],
                headers: [Header.authorization.rawValue:
authToken],
                responseType: .text,
                responseHandler: responseHandler)
    }

    // Запрос подборок конкурсов
    func getContestCompilations(with responseHandler: @escaping
(Bool, Int?, Any?) -> ()) {
        var authToken: String
        if let token = AccountManager.shared.authToken {
            authToken = token
        } else {
            authToken = ""
        }
        request(on: .get,
                for: .contestCompilations,
                params: [ParameterType.os.rawValue: "ios"],
                headers: [Header.authorization.rawValue: authToken],
                responseType: .text,
                responseHandler: responseHandler)
    }

    // Запрос подробной информации о подборке конкурсов
    func getCompilation(compilationId: Int, with responseHandler:
@escaping (Bool, Int?, Any?) -> ()) {
        var authToken: String
        if let token = AccountManager.shared.authToken {
            authToken = token
        } else {
            authToken = ""
        }
        request(on: .get,
                for: .contestCompilations,
                restfulId: compilationId,
                params: [ParameterType.os.rawValue: "ios"],
                headers: [Header.authorization.rawValue: authToken],
                responseType: .text,
                responseHandler: responseHandler)
    }

```

```

// Запрос проверки токена покупки подборки конкурсов
func validateContestCompilation(sku: String, token: String,
with responseHandler: @escaping (Bool, Int?, Any?) -> ()) {
    guard let authToken = AccountManager.shared.authToken else {
        return }
    request(on: .postData,
            for: .contestCompilationEnhancement,
            params: [ParameterType.os.rawValue: "ios",
                    ParameterType.sku.rawValue: sku,
                    ParameterType.purchaseToken.rawValue:
token],
            headers: [Header.authorization.rawValue: authToken],
            responseType: .text,
            responseHandler: responseHandler)
    }

// Запрос отзывов о подборке конкурсов
func compilationsReviews(compilationId: Int, responseHandler:
@escaping (Bool, Int?, Any?) -> ()) {
    var userToken = ""
    if let token = AccountManager.shared.authToken {
        userToken = token
    }
    requestURI(on: .get,
              for:
"contest_compilations/\(compilationId)/reviews",
              headers: [Header.authorization.rawValue:
userToken],
              responseType: .text,
              responseHandler: responseHandler)
    }

// Отправка/обновление нового отзыва
func postCompilationsReviews(_ action: CommentAction,
compilationId: Int, review: String, responseHandler: @escaping
(Bool, Int?, Any?) -> ()) {
    guard let token = AccountManager.shared.authToken else {
        return }
    let method: RequestType = action == .create ? .postData :
.put
    requestURI(on: method,
              for:
"contest_compilations/\(compilationId)/reviews",
              params: [ParameterType.comment.rawValue: review,
                      ParameterType.rating.rawValue: 0],
              headers: [Header.authorization.rawValue: token],
              responseType: .text,
              responseHandler: responseHandler)
    }

// Удаление отзыва

```

```

    func deleteCompilationReview(for compilationId: Int,
responseHandler: @escaping (Bool, Int?, Any?) -> ()) {
    guard let token = AccountManager.shared.authToken else {
return }
    requestURI(on: .delete,
                for:
"contest_compilations/\(compilationId)/reviews",
                headers: [Header.authorization.rawValue: token],
                responseHandler: responseHandler)
    }

}

extension ServerManager {
    static func errors(fromResponse response: Any?) -> [String]? {
    guard let response = response as? String,
        let data = response.data(using: .utf8),
        let json = try? JSONSerialization.jsonObject(with:
data, options: []) else { return nil }
    return ((json as? [String: Any])?["errors"] as?
String).flatMap { [$0] }
    }

    func showLogs(_ response: HTTPResult, ofType responseType:
ResponseType) {
    let r = responseType == .text ? response.text :
response.json
    print("-")
    if let response = response.response?.description {
    print(response)
    }
    print((r as? [String: Any])?["errors"] ?? "success")
    }
}

```

Приложение Г
РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ СЕРВЕРНОЙ ЧАСТИ
(рекомендуемое)

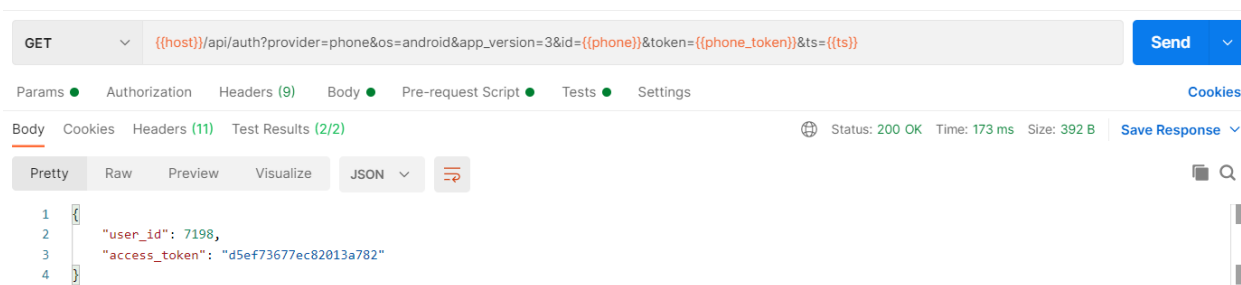


Рисунок Г.1 – Обращение к сервису входа в систему

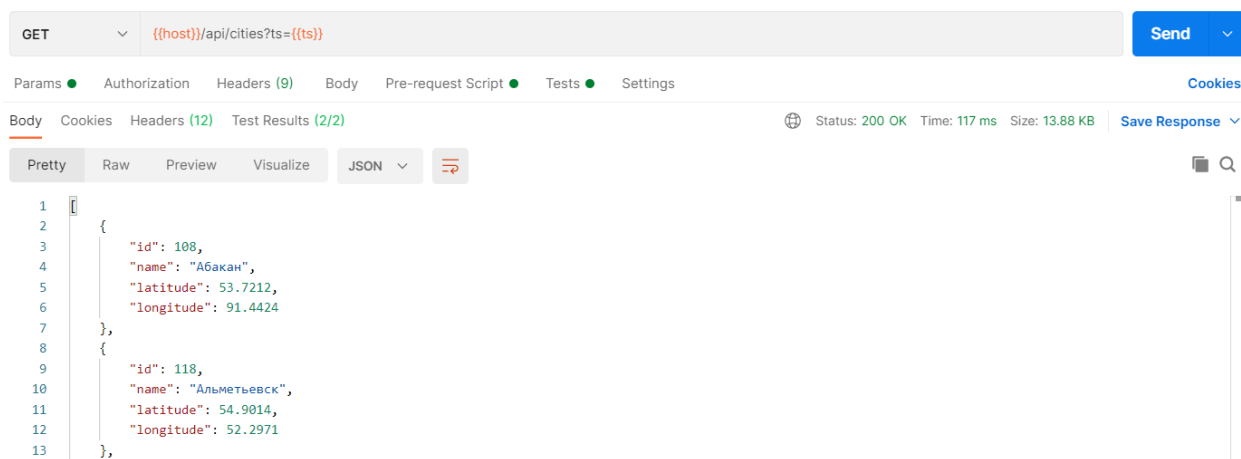


Рисунок Г.2 – Получение данных о городах

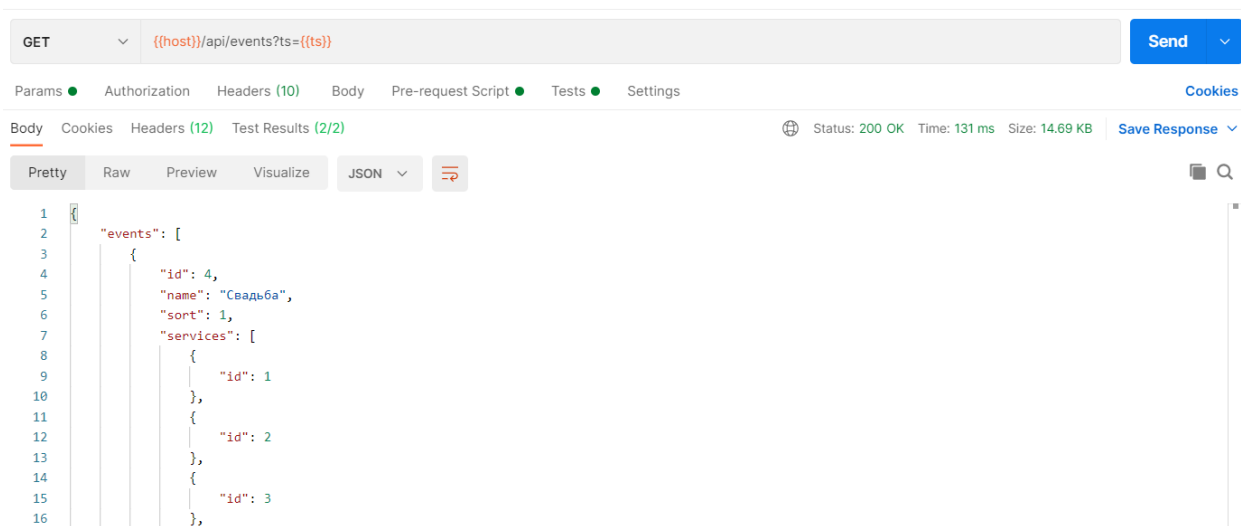


Рисунок Г.3 – Получение данных о типах конкурсов

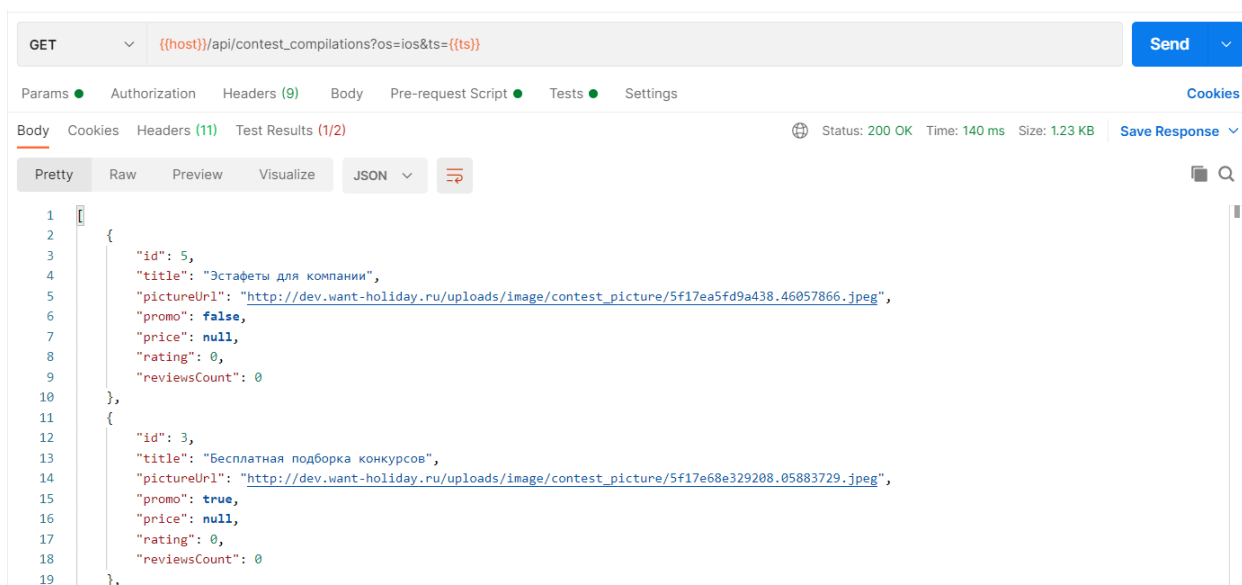


Рисунок Г.4 – Получение данных о конкурсных подборках

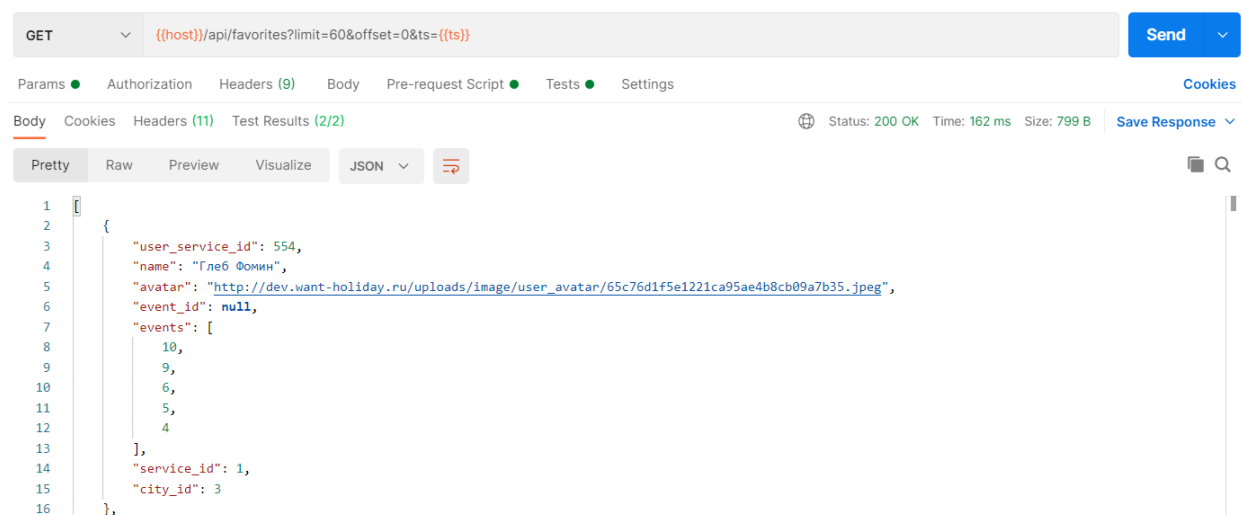


Рисунок Г.5 – Получение данных об избранных услугах

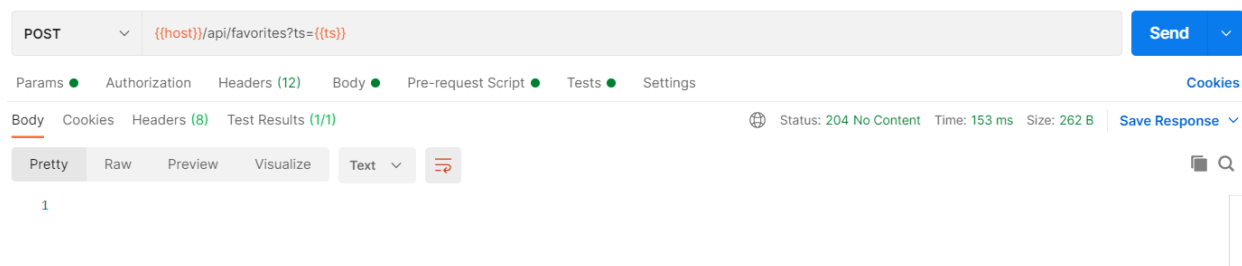


Рисунок Г.6 – Внесение услуги с номером 3479 в избранное

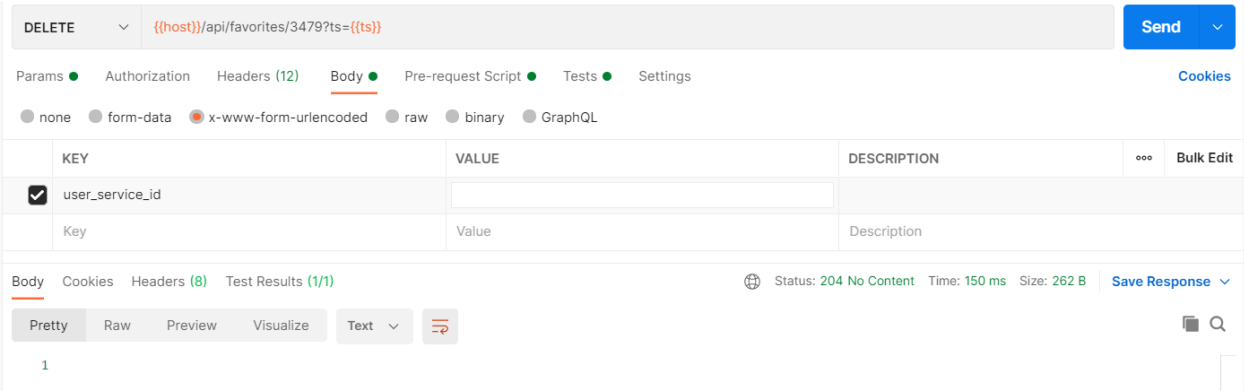


Рисунок Г.7 – Удаление избранной услуги с номером 3479

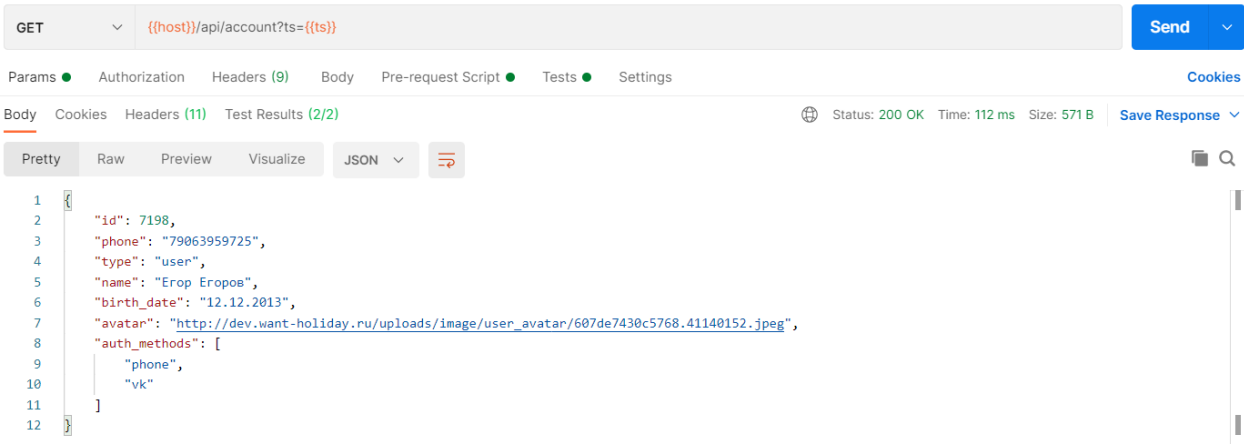


Рисунок Г.8 – Получение данных о пользователе

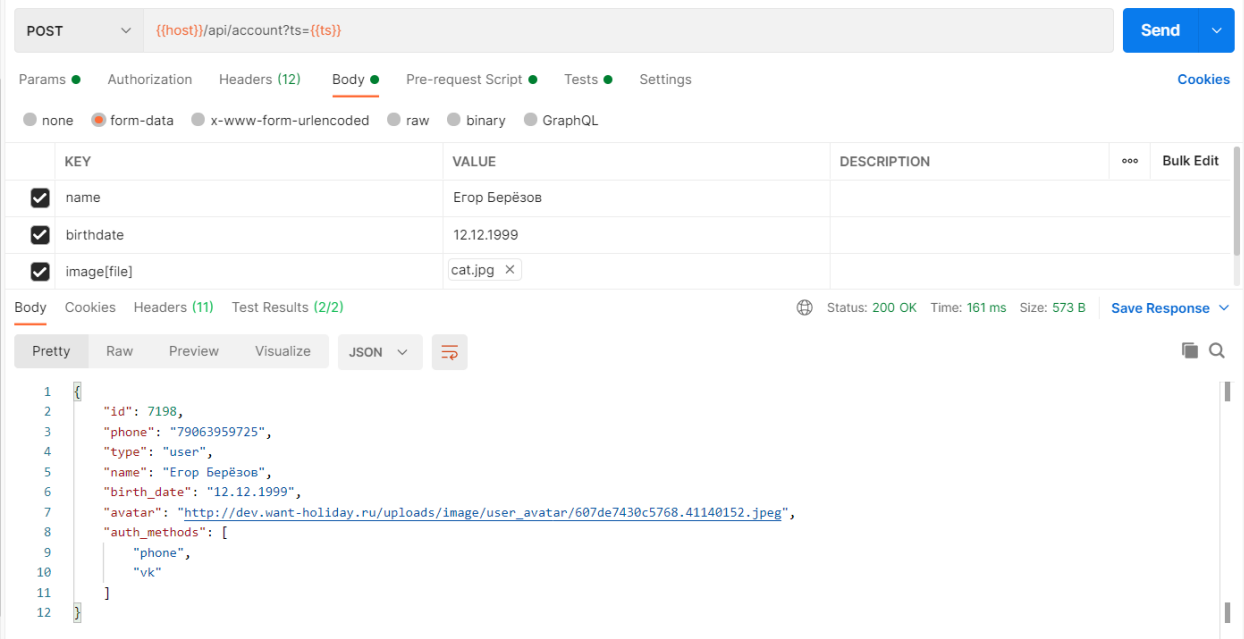


Рисунок Г.9 – Обновление данных о пользователе

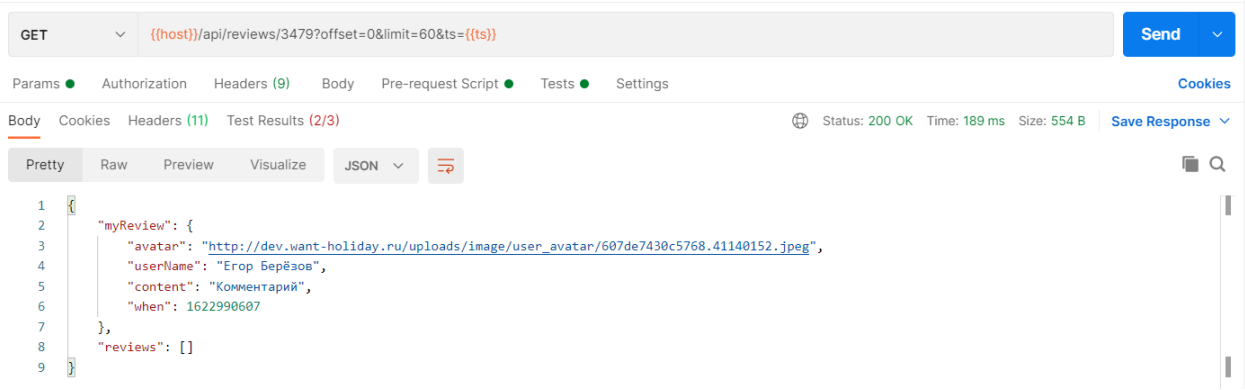


Рисунок Г.10 – Получение отзывов на услугу с номером 3479

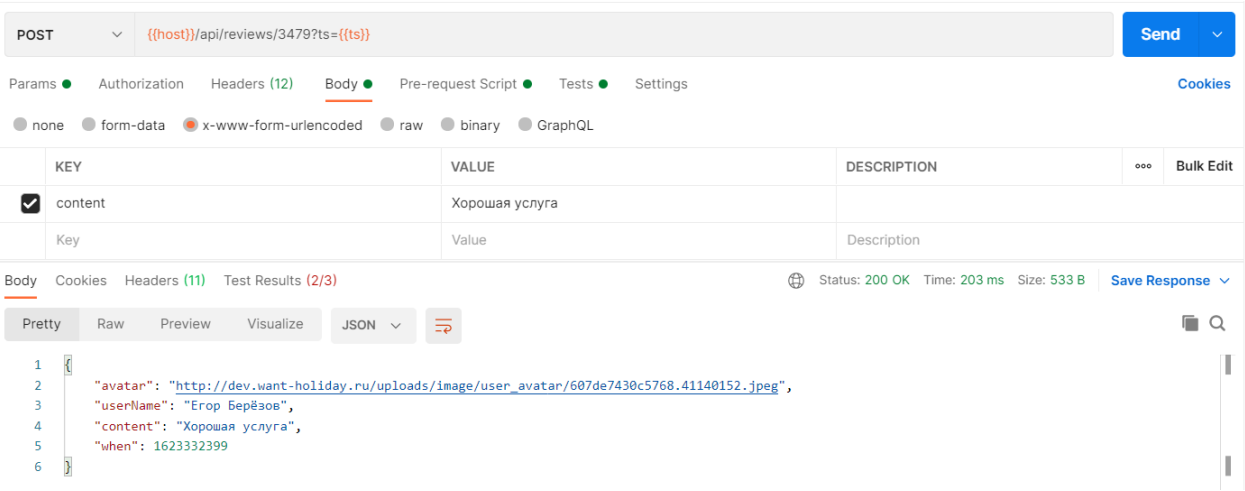


Рисунок Г.11 – Отправка нового отзыва на услугу с номером 3479

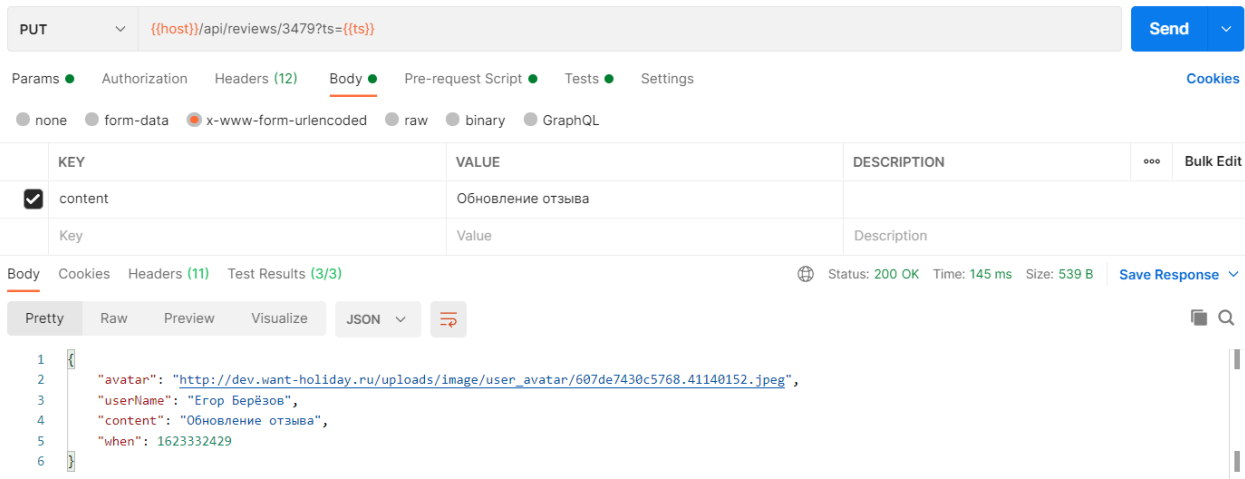


Рисунок Г.12 – Обновление отзыва на услугу с номером 3479

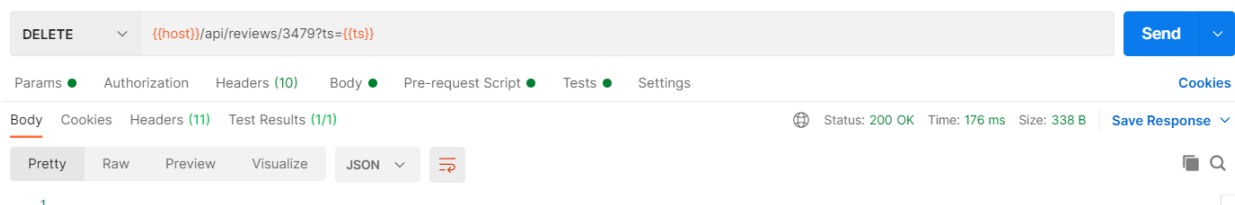


Рисунок Г.13 – Удаление отзыва на услугу с номером 3479

Приложение Д
РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ КЛИЕНТСКОЙ ЧАСТИ
(рекомендуемое)

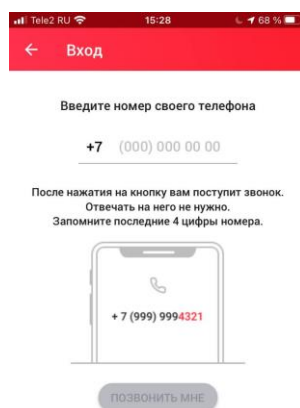


Рисунок Д.1 – Просмотр экрана «Вход»

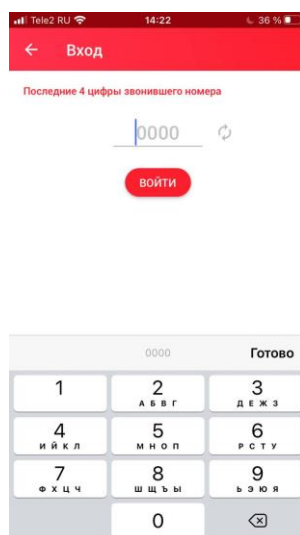


Рисунок Д.2 – Экран «Подтверждение входа»

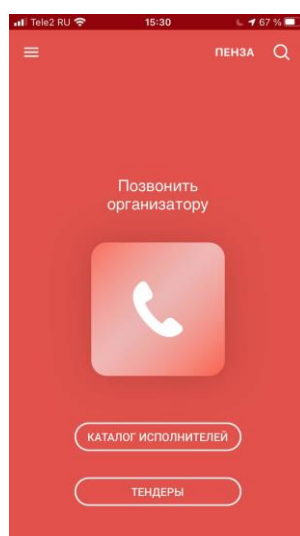


Рисунок Д.3 – Главный экран приложения



Рисунок Д.4 – Отображение сообщения об ошибке

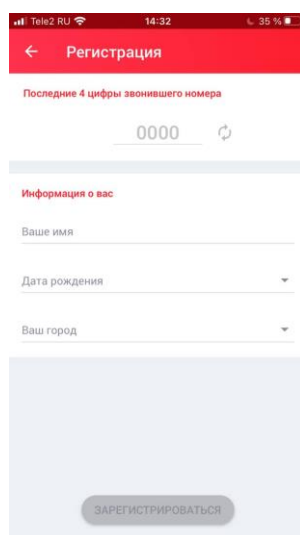


Рисунок Д.5 – Отображение экрана регистрации

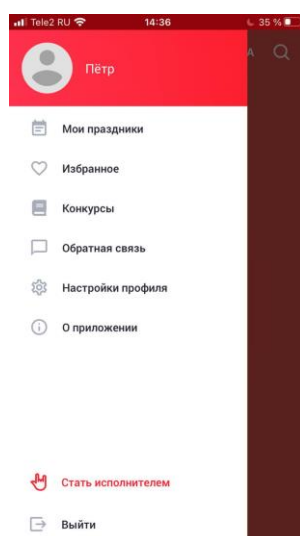


Рисунок Д.6 – Отображение бокового меню с именем пользователя

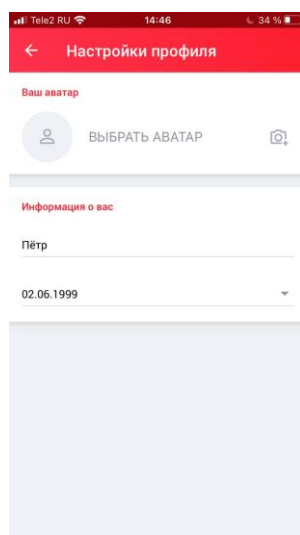


Рисунок Д.7 – Отображение настроек профиля

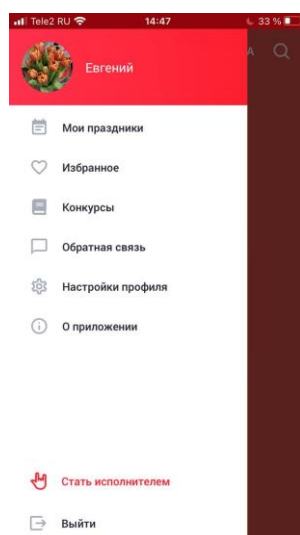


Рисунок Д.8 – Отображение изменений после редактирования профиля



Рисунок Д.9 – Отображение экрана «Каталог исполнителей»

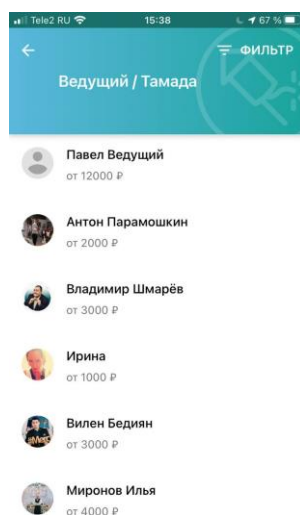


Рисунок Д.10 – Отображение списка в категории «Ведущий/Тамада»

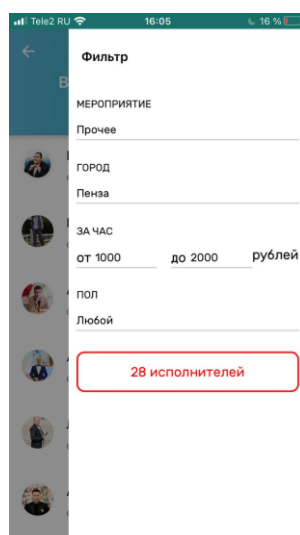


Рисунок Д.11 – Фильтры услуг

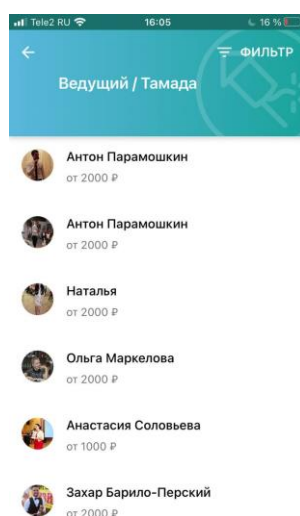


Рисунок Д.12 – Отфильтрованный список категории «Ведущий/Тамада»

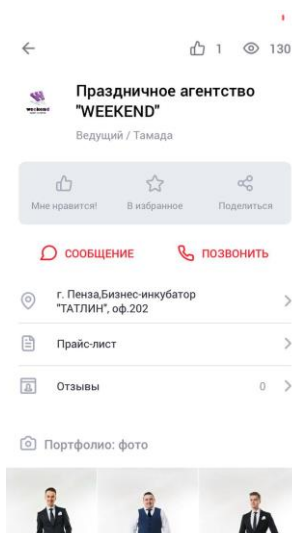


Рисунок Д.13 – Подробная информация об услуге

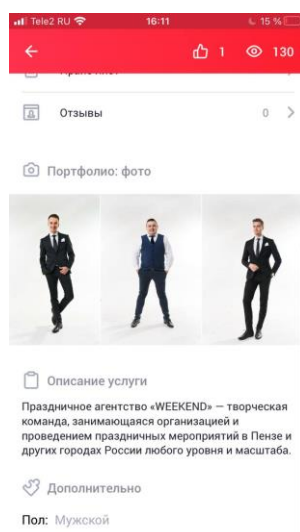


Рисунок Д.14 – Подробная информация об услуге

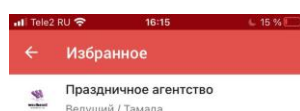


Рисунок Д.15 – Список избранных услуг

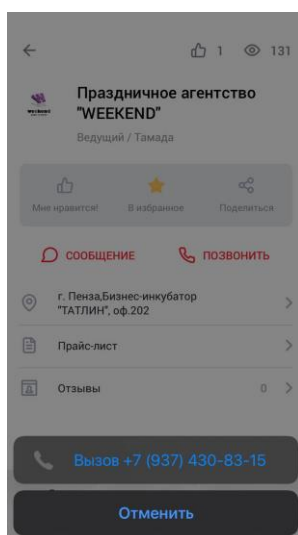


Рисунок Д.16 – Отображение представления вызова исполнителя

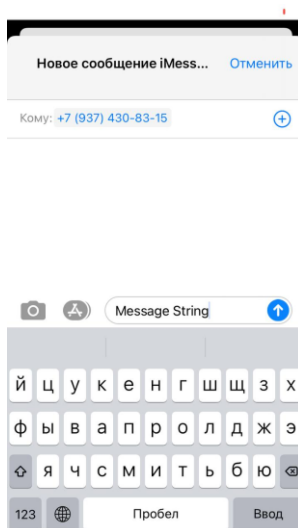


Рисунок Д.17 – Отображение диалога с исполнителем

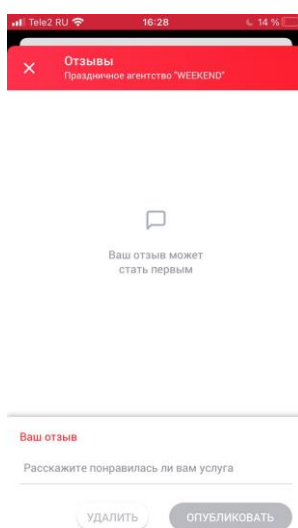


Рисунок Д.18 – Экран отзывов на услугу

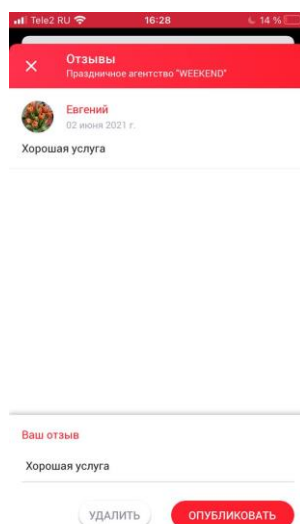


Рисунок Д.19 – Добавление отзыва на услугу

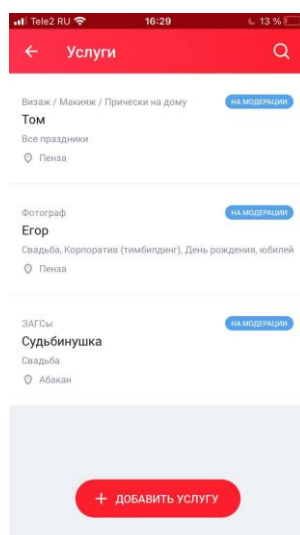


Рисунок Д.20 – Список созданных услуг



Рисунок Д.21 – Список возможных категорий при создании услуги

Рисунок Д.22 – Отображение экрана «Создание услуги»

Рисунок Д.23 – Обновленный список услуг

Рисунок Д.24 – Отображение экрана «Тендеры»

Рисунок Д.25 – Отображение экрана «Создание тендера»

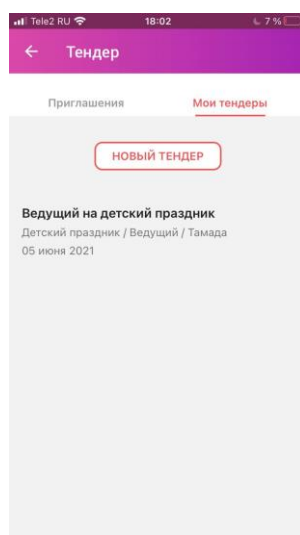


Рисунок Д.26 – Список с созданным ранее тендером

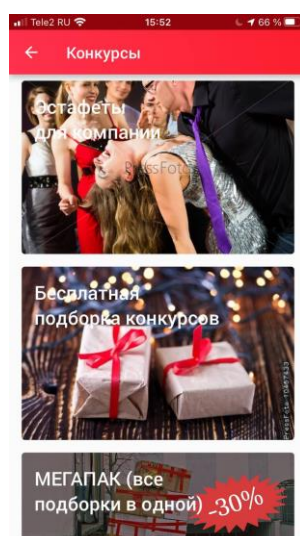


Рисунок Д.27 – Отображение экрана с списком конкурсных подборок



Рисунок Д.28 – Подробное описание конкурсной подборки



Рисунок Д.29 – Отображение экрана с купленной подборкой

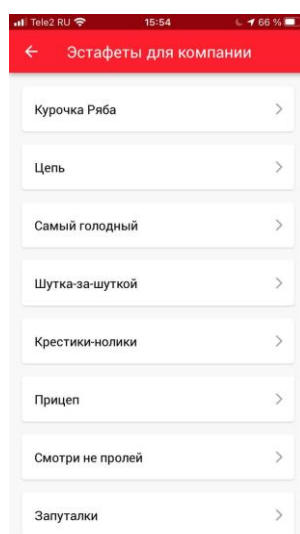


Рисунок Д.30 – Отображение экрана с списком конкурсов к подборке

Приложение Е
АКТ О ВНЕДРЕНИИ
(рекомендуемое)

АКТ

о внедрении результатов
выпускной квалификационной работы
Ермина Егора Александровича

Мобильное приложение поддержки процессов взаимного поиска потребителей и поставщиков социальных услуг, разработанное программистом Ерминым Е.А. – сотрудником ООО «СофтИнвент», в ходе выполнения выпускной квалификационной работы, успешно введено в эксплуатацию.

Генеральный директор

« 17 » июня 2021 г.

А.М. Княжев

