

Министерство науки и высшего образования Российской Федерации  
Санкт-Петербургский политехнический университет Петра Великого  
Институт компьютерных наук и технологий

Работа допущена к защите

Руководитель ОП

\_\_\_\_\_ В.Г. Пак

« \_\_\_\_ » \_\_\_\_\_ 2022 г.

## **ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

### **РАБОТА БАКАЛАВРА**

#### **РАЗРАБОТКА ПРОТОТИПА АВТОМАТИЗИРОВАННОЙ ИНФОРМАЦИОННОЙ СИСТЕМЫ УПРАВЛЕНИЯ ЗАКАЗАМИ НА ЭКСКУРСИИ ДЛЯ ЭКСКУРСИОННО-ОБРАЗОВАТЕЛЬНОГО ЦЕНТРА KID STREET**

по направлению подготовки 02.03.03 Математическое обеспечение и  
администрирование информационных систем

Направленность (профиль) 02.03.03\_01 Интеллектуальные информационные  
системы и обработка данных

Выполнил  
студент гр. 3530203/80102

Л.Д. Челищева

Руководитель  
ст. преподаватель ВШИИ

Е.Е. Андрианова

Консультант  
по нормоконтролю

Е.Е. Андрианова

**САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ  
ПЕТРА ВЕЛИКОГО**

**Институт компьютерных наук и технологий**

УТВЕРЖДАЮ

Руководитель ОП

В.Г. Пак

«    »                      2022 г.

**ЗАДАНИЕ**

**на выполнение выпускной квалификационной работы**

студентке Челищевой Лилии Дмитриевне, гр. 3530203/80102

1. Тема работы «Разработка прототипа автоматизированной информационной системы управления заказами на экскурсии для экскурсионно-образовательного центра Kid Street»
2. Срок сдачи студентом законченной работы: 05.2022.
3. Исходные данные по работе: сайт экскурсионно-образовательного центра Kid Street с описанием предоставляемых услуг, URL: <https://kid-street.ru/>.
4. Содержание работы (перечень подлежащих разработке вопросов):
  - 4.1. Анализ предметной области, существующих методов решения и средств разработки.
  - 4.2. Проектирование информационной системы управления заказами.
  - 4.3. Разработка прототипа пользовательского интерфейса, Web-API и базы данных.
  - 4.4. Развертывание прототипа информационной системы на сервере.
  - 4.5. Тестирование информационной системы.
5. Перечень графического материала (с указанием обязательных чертежей):
  - 5.1. Диаграмма прецедентов информационной системы.
  - 5.2. Диаграмма классов.
6. Консультанты по работе:  
Старший преподаватель, Е.Е. Андрианова (нормоконтроль).
7. Дата выдачи задания 03.02.2022.

Руководитель ВКР \_\_\_\_\_

Е.Е. Андрианова

Задание приняла к исполнению 03.02.2022.

Студент \_\_\_\_\_

Л.Д. Челищева

## РЕФЕРАТ

На 123 с., 1 кн., 24 рис., 1 табл., 21 источн., 5 прил.

**КЛЮЧЕВЫЕ СЛОВА:** ЭКСКУРСИОННЫЕ АГЕНТСТВА, ТУРИСТИЧЕСКИЕ ФИРМЫ, ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ, АВТОМАТИЗИРОВАННЫЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ, СИСТЕМЫ УПРАВЛЕНИЯ ВЗАИМООТНОШЕНИЯМИ С КЛИЕНТАМИ

Тема выпускной квалификационной работы: «Разработка прототипа автоматизированной информационной системы управления заказами на экскурсии для экскурсионно-образовательного центра Kid Street».

Целью выпускной квалификационной работы является проектирование и разработка прототипа АИС управления заказами на экскурсии для экскурсионно-образовательного центра КидСтрит.

В процессе работы была проанализирована структура предприятия, рассмотрены аналоги разрабатываемой АИС, определена необходимость разработки и выбран стек технологий. Далее было проведено проектирование структуры АИС, БД, Web API и пользовательского интерфейса. На основе проекта была выполнена разработка прототипа всех компонентов спроектированной АИС. Затем было проведено размещение серверных компонентов на хостинге, тестирование работы АИС и создание установочного пакета для пользовательского интерфейса.

В результате работы получен протестированный и полностью реализующий заявленный функционал прототип АИС, позволяющий управлять информацией о заказах в рамках работы экскурсионно-образовательного центра Kid Street. Полученный прототип готов к тестированию на производстве и доработке до релиз-версии в соответствии с полученной обратной связью.

## **ABSTRACT**

123 pages, 24 figures, 1 table, 5 appendices.

**KEYWORDS:** SIGHTSEEING AGENCIES, TRAVEL COMPANIES, SOFTWARE DESIGN, AUTOMATED INFORMATION SYSTEMS, CUSTOMER RELATIONSHIP MANAGEMENT SYSTEMS

The subject of the graduate qualification work is: «Development of a prototype of an automated information system for ordering excursions for the Kid Street Excursion and Education Center».

The purpose of the graduate qualification work is design and development of a prototype AIS for managing excursion orders for the Kid Street Excursion and Educational Center.

In the process of work, the structure of the enterprise was analyzed, analogues of the AIS being developed were examined, the need for development was determined and technology stack was selected. Further, the design of the AIS structure, database, Web API, and user interface was carried out. Based on the project, a prototype of all components of the designed AIS was developed. Then the server components were placed on the hosting, AIS operation was tested and an installation package for the user interface was created.

As a result of the work, a tested AIS prototype that fully implements the declared functionality was obtained, which allows managing information about orders within the framework of the Kid Street Excursion and Educational Center. The resulting prototype is ready for production testing and finalization to the release version in accordance with the feedback received.

## СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ.....	8
ВВЕДЕНИЕ.....	9
Глава 1 Анализ предметной области.....	11
1.1 Организационная структура предприятия.....	11
1.1.1 Управление заказами .....	11
1.1.2 Администрирование.....	13
1.2 Обзор подходов к реализации CRM.....	14
1.3 Обзор существующих решений.....	15
Глава 2 Обзор методов и средств разработки .....	17
2.1 Анализ методов решения.....	17
2.2 Сравнение и выбор технологии разработки клиентского приложения .....	18
2.2.1 Windows Forms .....	18
2.2.2 WPF.....	19
2.2.3 Вывод.....	19
2.3 Сравнение и выбор Web API.....	20
2.3.1 ASP.NET Web API.....	20
2.3.2 Laravel.....	20
2.3.3 Вывод.....	21
2.4 Сравнение и выбор СУБД .....	21
2.4.1 SQL Server .....	21
2.4.2 MySQL.....	21
2.4.3 Вывод.....	22

2.5 Обзор вспомогательных технологий.....	22
2.5.1 JSON .....	22
2.5.2 Entity Framework Core.....	23
Глава 3 Проектирование АИС .....	24
3.1 Общее проектирование АИС .....	24
3.2 Проектирование БД и Web API .....	26
3.3 Проектирование пользовательского интерфейса.....	28
Глава 4 Разработка и тестирование АИС.....	30
4.1 Разработка БД и Web API.....	30
4.2 Разработка пользовательского интерфейса .....	31
4.3 Тестирование АИС.....	39
ЗАКЛЮЧЕНИЕ .....	40
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ .....	41
ПРИЛОЖЕНИЕ А Диаграммы потоков данных АИС .....	43
ПРИЛОЖЕНИЕ Б SQL-скрипт БД.....	46
ПРИЛОЖЕНИЕ В Код основных элементов Web API .....	52
ПРИЛОЖЕНИЕ Г Код основных элементов пользовательского интерфейса .....	80
ПРИЛОЖЕНИЕ Д (Обязательное) Результаты тестирования АИС .....	111

## **ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ**

CRM (Customer Relationship Management) – это система, помогающая контролировать все каналы коммуникаций с клиентами и автоматизировать продажи [1]

DFD (Data Flow Diagrams) – диаграммы потоков данных

ER (Entity-Relationship) – сущность-связь

WPF – Windows Presentation Foundation

АИС – автоматизированная информационная система

БД – база данных

ВКР – выпускная квалификационная работа

ИС – информационная система

ПК – персональный компьютер

ПО – программное обеспечение

Стек технологий – это набор инструментов, применяющийся при работе в проектах и включающий языки программирования, системы управления базами данных, компиляторы и т. д. [2]

СУБД – система управления базами данных

Хостинг – это размещение интернет-проектов на физических и виртуальных серверах, подключенных к Интернету, для непрерывного обеспечения присутствия сайтов в Сети [3]

Журналирование — это процесс записи в хронологическом порядке событий, происходящих с каким-то объектом [4]



## **ВВЕДЕНИЕ**

На сегодняшний день в России, как и во многих других странах, туристическая отрасль является значимой частью экономики. Работа любой туристической фирмы предполагает в случае каждого заказа согласование большого количества разрозненных соглашений, и, следовательно, большой документооборот.

В связи с этим на рынке существует немало программных продуктов для поддержания и упрощения функционирования этой отрасли. Однако в большинстве случаев это комплексные дорогостоящие CRM, включающие в себя множество различных модулей, большинство из которых не используются данной конкретной компанией, при чем далеко не от всех из них можно отказаться при приобретении готового программного продукта или подписке на него.

Для небольших туристическим или экскурсионных агентств подобная избыточность напрямую вредна из-за заметно возрастающих излишних затрат и перегруженного ненужными элементами интерфейса. В связи с этим зачастую куда более эффективной выступает разработка компактной индивидуальной АИС с простым интерфейсом и функционалом, полностью соответствующим нуждам этой компании.

Исходя из вышеописанного, целью выпускной квалификационной работы выступает проектирование и разработка прототипа АИС управления заказами на экскурсии для экскурсионно-образовательного центра КидСтрит.

Задачами ВКР становится:

- Проанализировать структуру предприятия, рассмотреть существующие аналоги разрабатываемой АИС и обосновать необходимость создания нового программного продукта.
- Рассмотреть технологии и методы, теоретически применимые при разработке АИС и определить те из них, которые будут использованы при практической реализации.

- Спроектировать структуру АИС, БД, Web API и пользовательский интерфейс.
- Разработать БД, Web API и прототип пользовательского интерфейса.
- Разместить Web API и БД на тестовом хостинге.
- Протестировать разработанный программный продукт.
- Создать установочный пакет для пользовательского интерфейса.

## **Глава 1 Анализ предметной области**

### **1.1 Организационная структура предприятия**

Экскурсионно-образовательный центр КидСтрит является подразделением ООО «Мэйн Стрит» [5]. Данный центр оказывает экскурсионные услуги на территории Санкт-Петербурга и Ленинградской области, в первую очередь профориентационного характера.

Данная работа направлена на взаимодействие с отделом продаж, в задачу менеджеров которого входит обработка заявок на экскурсии и заключения договоров с клиентами. Также с системой будут работать администраторы, настраивающие редко изменяемые параметры экскурсий и списки сотрудников. Информация об экскурсиях, других услугах и работе экскурсоводов хранится у компании и может быть изменена.

#### **1.1.1 Управление заказами**

Любой менеджер при поступлении новой заявки от клиента должен сформировать заказ на экскурсию. Формирование заказа включает следующие этапы:

- 1) Получение заявки;
- 2) Оформление заказа;
- 3) Подсчет стоимости заказа;
- 4) Утверждение заказа у клиента;
- 5) (Опционально) внесение правок в заказ;
- 6) Оформление договора (в данный пункт входит внесение предоплаты клиентом);
- 7) Подтверждение предоплаты.

После внесения предоплаты заказ можно только отменить, но не отредактировать. Также после предоплаты заказчик получает сводный отчет по заказу.

В информацию о заказе входят следующие параметры:

- Наименование экскурсии;
- Название образовательного учреждения или иной организации или группы экскурсантов;
- Возрастная категория и (если требуется) школьный класс;
- Количество участников и сопровождающих;
- Контактные данные заказчика (телефон, электронная почта, (опционально) адрес в социальной сети ВК, его отношение к ученикам (учитель/родитель));
- Дата и время мероприятия;
- Место встречи;
- Экскурсовод;
- Стоимость;
- Комментарии;
- Дополнительные услуги;
- Статус.

Основным мероприятием тура может выступать только одна экскурсия, однако существует ряд дополнительных услуг, в том числе и иные места посещения.

Заказ имеет следующие ограничения:

- Суммарное время всех мероприятий не может превышать 8 часов с учетом всех дополнительных услуг с указанным временем.
- Группу до 15 детей включительно должен сопровождать по крайней мере один взрослый, группу от 16 детей должны сопровождать двое взрослых.
- Группа от 35 детей требует отдельного согласования и корректировки стоимости заказа.

В информацию об экскурсиях входят:

- название;
- тип (экскурсия, мастер-класс, квест);

- направление;
- продолжительность;
- описание и условия;
- дни проведения (любые дни недели в любых комбинациях);
- возрастная группа (5-6 лет, 1-11 классы, студенты, взрослые – в любых комбинациях);
- цены.

У любой экскурсии может быть несколько вариантов цен в зависимости от числа участников.

Дополнительные услуги в заказе имеют только наименование и стоимость. В случае включения их в заказ так же указывается их количество.

### **1.1.2 Администрирование**

В рамках администрирования в системе хранится и модифицируется следующая информация о сотрудниках (менеджерах, администраторах и экскурсоводах):

- паспортные данные;
- полное имя;
- контактный телефон.

Для экскурсоводов также хранится список экскурсий, которые они могут вести.

Администратор отвечает за обновление вышеуказанных данных.

Кроме того, администраторам компании требуются следующие статистические данные:

- Процентное соотношение различных возрастных групп среди заказов;
- Распределение числа заказов по месяцам.

Должна быть возможность отследить, кто именно вносил изменения в данные.

На данный момент сбор заявок автоматизирован системой amoCRM,

остальные же этапы выполняются вручную с использованием офисных пакетов.

## **1.2 Обзор подходов к реализации CRM**

Как уже было отмечено ранее, на рынке для экскурсионных фирм существует множество готовых решений для автоматизации и упрощения работы с заказчиками, чаще всего являющихся CRM. Подобное программное обеспечение может выполнять целый ряд задач, включающих:

- Собирать, хранить и обрабатывать данные о клиентах;
- Собирать, хранить и обрабатывать данные о компании, включая данные о сотрудниках;
- Обеспечивать взаимодействие с клиентами;
- Координировать работу других бизнес-приложений;
- Автоматизировать повседневные рабочие задачи;
- Генерировать аналитические данные. [6]

Все CRM подразделяются на три вида в соответствии с выполняемыми ими задачами:

1) CRM-системы для совместной работы – главной задачей таких систем является устранение разрозненности. Они гарантируют, что все команды имеют доступ к одним и тем же актуальным данным о клиентах, независимо от того, в каком отделе или канале они работают.

2) Операционные CRM-системы – помогают оптимизировать процессы компании для взаимоотношений с клиентами. Они предоставляют инструменты для лучшей визуализации и более эффективного управления полным циклом взаимодействия с клиентом. Операционные CRM-системы обычно предоставляют функции автоматизации.

3) Аналитические CRM-системы – позволяют анализировать собранную иными способами информацию о клиентах.

Выбор необходимого типа или типов CRM для компании зависит от множества факторов, однако далеко не всегда нужны сразу все три ПО или

только один. Если бизнес новый и еще мало собранных данных о клиентах, аналитическая CRM может быть излишней. Потребность в CRM для совместной работы особенно актуальна, когда в компании много отделов или разных бизнес-центров. Операционная CRM наиболее важна для компаний, стремящихся улучшить процессы, связанные с полным жизненным циклом клиента, и для тех, кто хочет использовать автоматизацию для повышения эффективности.

Для небольших компаний зачастую требуется только часть функций каждого типа CRM, и, следовательно, в данной работе представлена гибридная система, включающая задачи операционного и аналитического типа.

### **1.3 Обзор существующих решений**

Для определения необходимости разработки нового программного продукта следует рассмотреть наиболее популярные CRM, применимые к экскурсионным агентствам, и выделить их основные особенности.

Самым популярным и распространённым на рынке выступает ПК «САМО-тургид». Данная комплексная система объединяет в себе бэк-офис, онлайн-бронирование для туристов и партнеров. Основное предназначение – создание любого вида экскурсионного маршрута или услуг, продажа их клиентам в режиме онлайн через сайт, ведение внутреннего управления и учета, а также взаимодействие с гидами и партнерами. [8] Цена данного программного продукта варьируется от 400 до 550 тысяч рублей.

Также широкое распространение на рынке имеет программа «Экскурсии.BidBir», позволяющая автоматически принимать заявки, формировать расписание, посчитать количество свободных мест и подготовить расчеты с партнерами. [9] Цена данного программного обеспечения варьируется в зависимости от количества заказов и составляет 5 рублей с каждого человека в каждом заказе.

Наиболее гибкой и настраиваемой средой выступает ExaExcursions – база данных, содержащая в себе всю информацию об экскурсиях, точках

продаж и точках выезда, о кассирах и гидах, водителях и экскурсоводах, а также о БСО, транспорте, местах в автобусах, трансферах, зарплатах и ряде других возможностей в режиме реального времени через сеть Интернет. [10] Благодаря модульной структуре цена за программный продукт может варьироваться от 25 до 800 тысяч рублей, однако следует отметить, что без большого количества подключенных платных модулей программа крайне малофункциональна, в то же время большая часть модулей предоставляет в одном комплекте сразу широкий спектр возможностей.

Проанализировав существующие решения, можно заключить, что даже при наличии готовых, даже модульных решений, остается необходимость в разработке узконаправленного программного продукта для обеспечения нужд автоматизации и модернизации работы небольших экскурсионных компаний в лице экскурсионно-образовательного центра КидСтрит. Данная необходимость возникает в связи с высокой стоимостью существующих CRM, частично связанной с избыточным функционалом, от которого нет возможности отказаться. В то же время лишние модули повышают нагрузку на систему, увеличивая энергозатраты и требования к оборудованию.



## **Глава 2 Обзор методов и средств разработки**

### **2.1 Анализ методов решения**

Как уже отмечено в подразделе 1.1.2, до начала данной работы в фирме КидСтрит большая часть задач выполнялась вручную. Помимо получения данных от агрегатора заявок, максимальный уровень автоматизации обеспечивался за счет использования калькулятора и приложения MS Office Excel. Такой подход обеспечивал крайне низкую эффективность и при этом высокий шанс ошибки. Кроме того, у каждого менеджера были свои источники данных, которые могли подвергаться изменениям независимо друг от друга.

Подобный вопрос решается несколькими способами, а наиболее радикальным является полная автоматизация процесса обработки заказов, например, посредством системы, принимающей заявки через агрегатор и на основе пожеланий клиентов с использованием методов машинного обучения формирующей подходящий план экскурсии.

Однако подобный способ, в первую очередь, практически полностью исключает менеджера из процесса оформления тура, ставя под сомнение необходимость в этой должности, что не являлось целью работы. Кроме того, далеко не все услуги и возможные параметры тура статичны и могут быть рассчитаны автоматически – спецификой сферы услуг, к которой относятся и экскурсионные услуги, является высокое влияние человеческого фактора [11], так что в ряде вопросов необходимо участие оператора.

В итоге для решения задачи была выбрана автоматизированная информационная система. В связи с необходимостью поддерживать общую базу данных для нескольких сотрудников при отсутствии локальной сети предприятия, и при этом необходимости минимизировать размер клиентского приложения в расчете на слабые ПК, была выбрана структура АИС в виде «Клиент – Сервер – БД». В данной схеме клиентом должен выступить пользовательский интерфейс, устанавливаемый для каждого из сотрудников

отдельно при необходимости, а сервер в виде Web API и БД должны размещаться на стороннем хостинге.

В качестве технологий для пользовательского интерфейса были выбраны технологии .NET и язык C#, как одни из наиболее стабильно развивающихся и широко применяемых. Процесс выбора отдельных технологий для различных компонентов АИС приведен в подразделах 2.2-2.4, при чем учитывается тот факт, что на ПК в организации установлена операционная система Windows 10.

## **2.2 Сравнение и выбор технологии разработки клиентского приложения**

### **2.2.1 Windows Forms**

Windows Forms – это полнофункциональный клиентский API, впервые появившийся в первой версии .NET Framework в 2000 году. По сравнению с WPF, рассмотренном в пункте 2.2.2, Windows Forms – относительно простая технология, предоставляющая большинство функций, необходимых для написания типичного приложения Windows. Она также имеет большое значение для поддержки устаревших приложений. Но по сравнению с WPF у него есть некоторое количество недостатков, большинство из которых связано с тем, что он является оболочкой над GDI+ и библиотекой Win32:

- API для отрисовки нестандартных элементов управления – GDI+, который, хотя и является достаточно гибким, медленно отрисовывает большие области (и без двойной буферизации может мерцать).
- Элементам управления не достает истинной прозрачности.
- Большинство элементов управления являются некомпозиционными.

С положительной стороны, Windows Forms относительно прост в использовании и имеет большое количество сторонних элементов управления и готовых библиотек. [12]

### **2.2.2 WPF**

WPF, как и Windows Forms являющийся технологией .NET Framework, был представлен в 2006 году и с тех пор постоянно совершенствуется. В отличие от своего предшественника, Windows Forms, WPF явно отображает элементы управления с помощью DirectX со следующими преимуществами:

- Поддержка сложной графики (произвольное преобразование, 3D-рендеринг, мультимедиа и прозрачность).
- Размерность на основе пикселей.
- Гибкая поддержка компоновки.
- Использование DirectX позволяет ускорить рендеринг и использовать аппаратное ускорение графики.
- Поддерживает надежную привязку данных.
- Внешний вид отделен от функциональности. [12]

В то же время следует отметить снижение темпов развития данной технологии и переключении создателей на другие направления. В последних версиях .NET более обширное внимание уделено обновлению, например, Windows Forms, что вызывает сомнения в перспективах технологии WPF в будущем.

### **2.2.3 Вывод**

В связи с необходимостью разработать простой, не ресурсозатратный интерфейс без использования сложной графики для решения поставленной задачи была выбрана технология Windows Forms на платформе .NET 5 для поддержания стабильного функционирования и максимальных возможностей технологии. для ведения разработки был выбран язык C# как наиболее распространенный для данной технологии.

## **2.3 Сравнение и выбор Web API**

### **2.3.1 ASP.NET Web API**

ASP.NET Core – облегченный модульный преемник .NET Framework с поддержкой популярного шаблона Model-View-Controller. ASP.NET Core используется для создания веб-сайтов, веб-API на основе REST и микросервисов.

ASP.NET Core работает в Windows, Linux и macOS и может самостоятельно размещаться в пользовательском процессе.

Как и любая архитектура тонкого клиента, ASP.NET Core предлагает следующие общие преимущества по сравнению с полнофункциональными клиентами:

- Нет развертывания на стороне клиента.
- Клиент может работать на любой платформе, поддерживающей веб-браузер.
- Обновления легко развертываются. [12]

### **2.3.2 Laravel**

Laravel – это бесплатный PHP фреймворк с открытым исходным кодом, имеющий ряд особенностей, в том числе:

- С помощью менеджера пакетов, фреймворк Laravel позволяет легко устанавливать и подключать различные компоненты для использования в веб-приложении.
- Механизм автозагрузки классов позволяет не подключать вручную файлы через `include` и предотвращает загрузку неиспользуемых компонентов.
- Система миграций помогает упростить развертывание и обновление веб-приложения.
- При создании приложения можно использовать Artisan – интерфейс командной строки для ввода встроенных команд, а также создания

своих собственных. [13]

### **2.3.3 Вывод**

В связи с выбором на предыдущем этапе в пункте 2.2.3 технологии модульной платформы .NET Windows Forms и языка С#, наиболее эффективным вариантом является использование технологии той же платформы – Web API на основе ASP.NET Core. Данный выбор позволит обеспечить взаимодействие клиентской части и Web API с наименьшими затратами и преобразованиями.

## **2.4 Сравнение и выбор СУБД**

### **2.4.1 SQL Server**

Microsoft SQL Server – СУБД, разработанная корпорацией Microsoft.

Основные особенности MS SQL:

- Основной используемый язык запросов – Transact-SQL. Transact - SQL позволяет использовать дополнительный синтаксис для хранимых процедур и обеспечивает поддержку транзакций.
- Microsoft SQL Server и Sybase ASE для взаимодействия с сетью используют протокол уровня приложения под названием Tabular Data Stream (TDS, протокол передачи табличных данных).
- Microsoft SQL Server также поддерживает Open Database Connectivity (ODBC) – интерфейс взаимодействия приложений с СУБД.
- SQL Server поддерживает зеркалирование и кластеризацию баз данных. [14]

### **2.4.2 MySQL**

MySQL – СУБД, разработанная корпорацией Oracle.

Программное обеспечение MySQL – ПО с открытым кодом. Доступно также большое количество программного обеспечения MySQL,

разработанного сторонними разработчиками.

Основные особенности MySQL:

- Работа на различных операционных системах.
- Многопоточность с использованием потоков ядра.
- Дисковые таблицы на основе В-деревьев со сжатием индексов.
- Используется оптимизированный метод однопроходного мультисоединения.
- Как временные таблицы используются в памяти хеш-таблицы.
- SQL-функции реализованы при помощи хорошо оптимизированной библиотеки классов. [15]

Недостатки и ограничения MySQL: не реализована поддержка транзакций, проблемы с надежностью – из-за некоторых способов обработки данных MySQL (связи, транзакции, аудиты) иногда уступает другим СУБД по надежности. [16]

### **2.4.3 Вывод**

Вследствие выбора технологии ASP.NET Web API, выполненного в пункте 2.3.3, для поддержания серверной части приложения на стороннем хостинге требуется, чтобы данный хостинг поддерживал C# как язык сервера вместо PHP. Подобные хостинги крайне редко предоставляют СУБД MySQL и в подавляющем большинстве предоставляют возможность работы только с Microsoft SQL Server. Поскольку с данной ситуации выбор в пользу MySQL породит потенциальные проблемы для клиента с выбором места размещения серверной части, было принято решение использовать СУБД Microsoft SQL Server.

## **2.5 Обзор вспомогательных технологий**

### **2.5.1 JSON**

JSON (JavaScript Object Notation) - простой формат обмена данными,

удобный для чтения и написания как человеком, так и компьютером. Он основан на подмножестве языка программирования JavaScript. JSON - текстовый формат, полностью независимый от языка реализации, но он использует соглашения С-подобных языков. Эти свойства делают JSON идеальным языком обмена данными.

JSON основан на двух структурах данных:

- Коллекция пар ключ/значение. В разных языках эта концепция реализована как объект, запись, структура, словарь, хэш, именованный список или ассоциативный массив.
- Упорядоченный список значений. В большинстве языков это реализовано как массив, вектор, список или последовательность.

Это универсальные структуры данных. Почти все современные языки программирования поддерживают их в какой-либо форме. [17]

### **2.5.2 Entity Framework Core**

Entity Framework Core представляет собой объектно-ориентированную, легковесную и расширяемую технологию от компании Microsoft для доступа к данным. EF Core позволяет работать с базами данных, но представляет собой более высокий уровень абстракции: EF Core позволяет абстрагироваться от самой базы данных и ее таблиц и работать с данными независимо от типа хранилища. На концептуальном уровне, который предлагает Entity Framework, производится работа с объектами.

Entity Framework Core поддерживает множество различных систем баз данных. Таким образом, через EF Core возможно работать с любой СУБД, если для нее существует нужный провайдер.

По умолчанию на данный момент Microsoft предоставляет ряд встроенных провайдеров: для работы с MS SQL Server, для SQLite, для PostgreSQL. Также имеются провайдеры от сторонних поставщиков, например, для MySQL. [18]

## **Глава 3 Проектирование АИС**

### **3.1 Общее проектирование АИС**

В ходе изучения работы отдела по работе с клиентами экскурсионно-образовательного центра КидСтрит были выделены следующие требующие автоматизации направления:

- Управление экскурсиями;
  - 1) Создание экскурсии;
  - 2) Изменение экскурсии;
  - 3) Удаление экскурсии;
  - 4) Создание цены экскурсии;
  - 5) Изменение цены экскурсии;
  - 6) Удаление цены экскурсии;
- Управление дополнительными услугами;
  - 1) Добавление дополнительной услуги;
  - 2) Изменение дополнительной услуги;
  - 3) Удаление дополнительной услуги;
- Управление заказами;
  - 1) Добавление заказа;
  - 2) Изменение заказа;
  - 3) Утверждение заказа у клиента;
  - 4) Подтверждение предоплаты;
  - 5) Подтверждение выполнения
  - 6) Удаление заказа;
- Создание отчетной документации;
  - 1) Отчет о соотношении различных возрастных групп среди заказов;
  - 2) Отчет о распределении числа заказов по месяцам;
- Управление сотрудниками;
  - 1) Добавление нового сотрудника;
  - 2) Изменение данных о сотруднике;



- 3) Удаление сотрудника;
- Управление типами экскурсий;
- 1) Добавление типа;
- 2) Изменение типа;
- 3) Удаление типа;
- Управление направлениями экскурсий;
- 1) Добавление направления;
- 2) Изменение направления;
- 3) Удаление направления;

Сформированные направления представлены на диаграмме вариантов использования на рисунке 1. При этом было учтено логическое разделение системы на два модуля – модуля администрирования и модуля управления заказами (менеджмента).

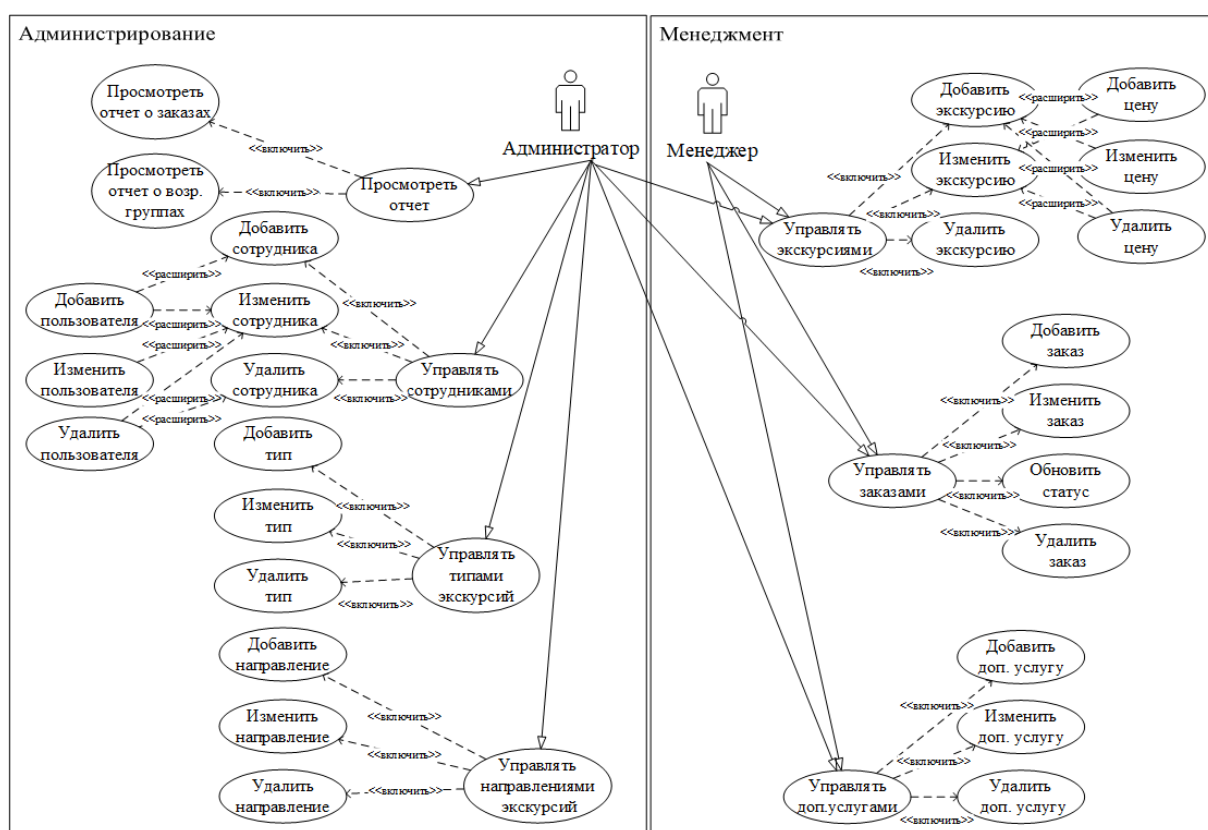


Рисунок 1 – Диаграмма вариантов использования

В рамках проектирования АИС была так же визуализирована подробная структура проектируемой АИС, представленная в приложении А в виде DFD на рисунках А.1, А.2, А.3 и А.4.

### 3.2 Проектирование БД и Web API

В ходе последующего анализа были выделены основные сущности проектируемой БД и определены их поля. Список сущностей включает:

- Тип экскурсии;
- Направление;
- Заказ;
- Статус заказа;
- Сотрудник;
- Экскурсия;
- Пользователь;
- Возрастная группа;
- Дни проведения;
- Цена экскурсии;
- Дополнительная услуга в заказ;
- Дополнительная услуга.

Выделенные сущности были визуализированы в виде логической ER-диаграммы, которая представлена на рисунке 2. На основе анализа сущностей была составлена итоговая физическая ER-диаграмма, в которой были подобраны имена таблиц и полей, а также типы данных и ограничения. Данная диаграмма представлена на рисунке 3. Типы данных выбирались с учетом СУБД SQL Server 2019 [14].

Кроме того, было принято решение, что для сохранения целостности данных любые процессы удаления записей, имеющих связи с подчиненными таблицами, должны быть выполнены через перемещение ненужных данных в архив (отметка как устаревших). Устаревшие данные можно применять только в тех подчиненных записях, где они уже включены, но нельзя добавлять в новые. Также в связи с введением подвидов экскурсии далее в тексте ВКР называются мероприятиями или услугами.

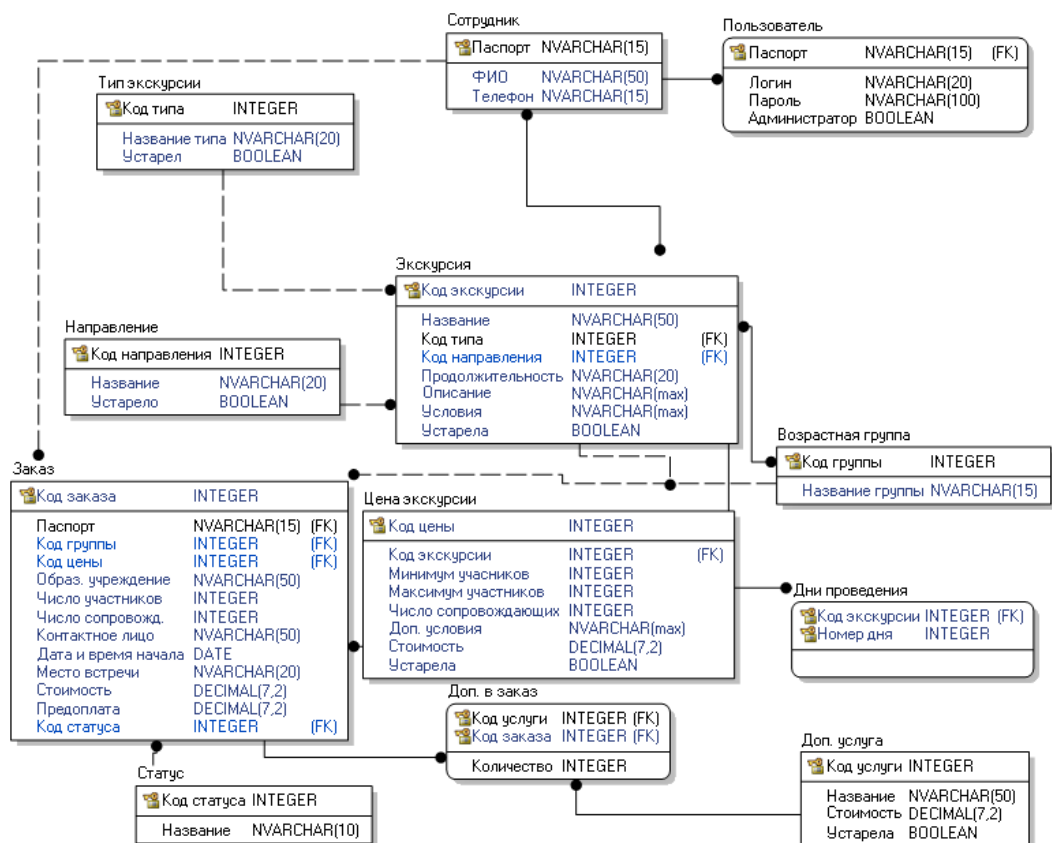


Рисунок 2 – Логическая ER-диаграмма

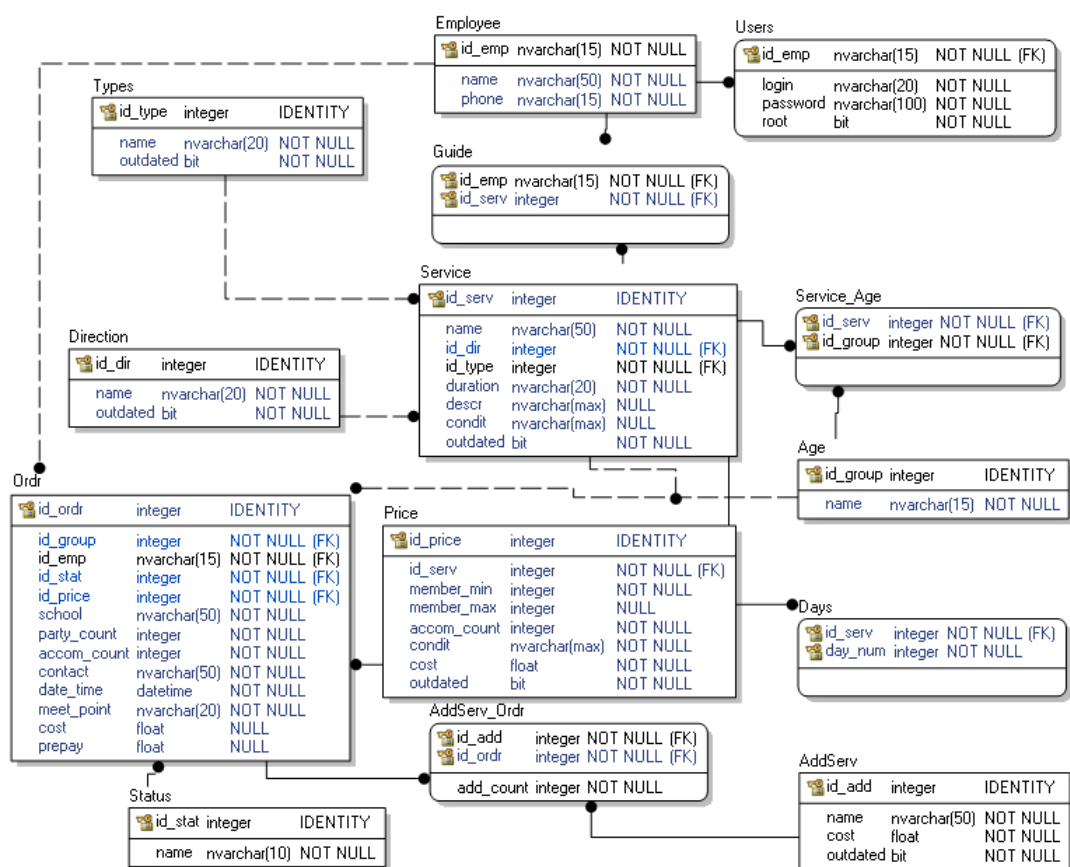


Рисунок 3 – Физическая ER-диаграмма

В рамках проектирования серверной части АИС была определена структура Web API [19], являющееся вариацией структуры Model-View-Controller. Предполагается использование общего класса контроллеров, разделенного на 4 группы задач:

- Контроллер вывода данных.
- Контроллер добавления данных и авторизации.
- Контроллер изменения данных.
- Контроллер удаления данных.

Все функции контроллера, отвечающие за модификации основных таблиц, должны вызывать процесс журналирования, описанный во вспомогательной функции.

Так же должна присутствовать модель, описывающая все таблицы БД в соответствии с типами данных и именами, представленными в физической ER-диаграмме на рисунке 3.

Компоненты представления из схемы Model-View-Controller в Web API не включаются из-за отсутствия необходимости.

### **3.3 Проектирование пользовательского интерфейса**

На основе анализа предметной области был определен основной функционал приложения (рисунок 1) и, впоследствии, экранные формы (окна), необходимые для выполнения этого функционала. Полная структура экранных форм представлена на рисунках 4 и 5.

Клиентское приложение включает в себя два модуля: менеджмент и администрирование, попадание в каждый из которых выполняется путем авторизации в системе. Каждый из модулей на этапе прототипа должен включать в себя собственное главное окно, содержащее ссылки на окна, отвечающие за выполнение конкретных задач. Все задачи делятся на группы:

- Вывод списка данных.
- Модификация (добавление, изменение, удаление) данных.
- Просмотр подробных данных по записи.
- Отчеты о данных в системе.

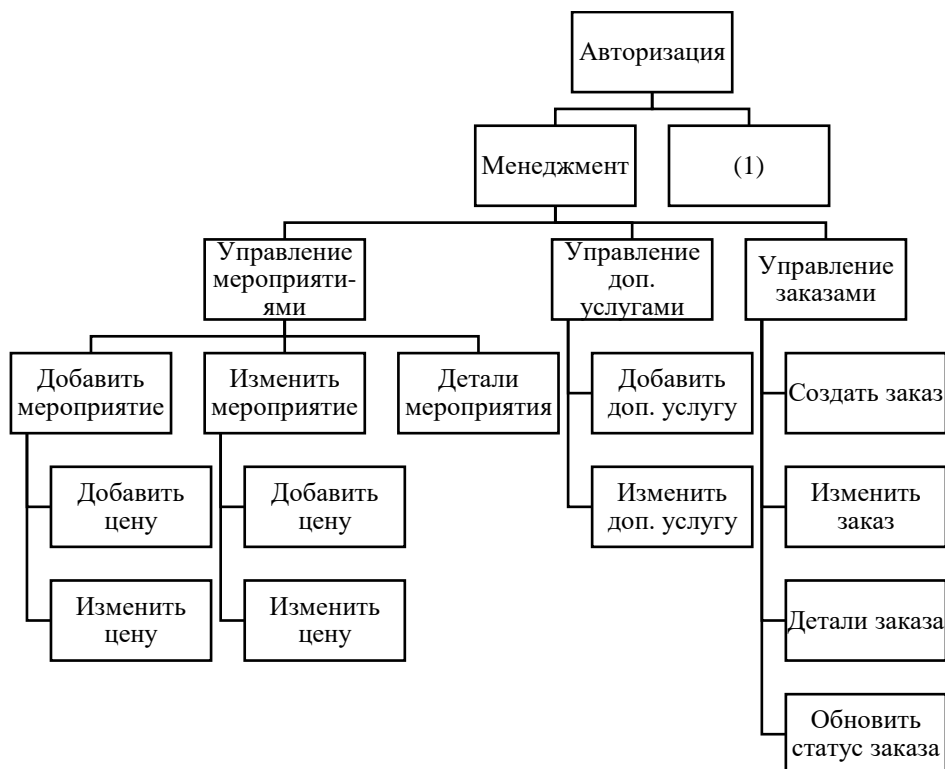


Рисунок 4 – Иерархия экранных форм авторизации и менеджмента



Рисунок 5 – Иерархия экранных форм администрирования

## Глава 4 Разработка и тестирование АИС

### 4.1 Разработка БД и Web API

На основе физической ER-диаграммы (рисунок 3) был создан SQL-скрипт базы данных, включающий все указанный на схеме сущности. Скрипт представлен полностью в приложении Б, включая заполнение таблиц тестовыми данными.

Данная БД была размещена на сервере тестового хостинга <https://somee.com/> в соответствии с параметрами, указанными на рисунке 6.

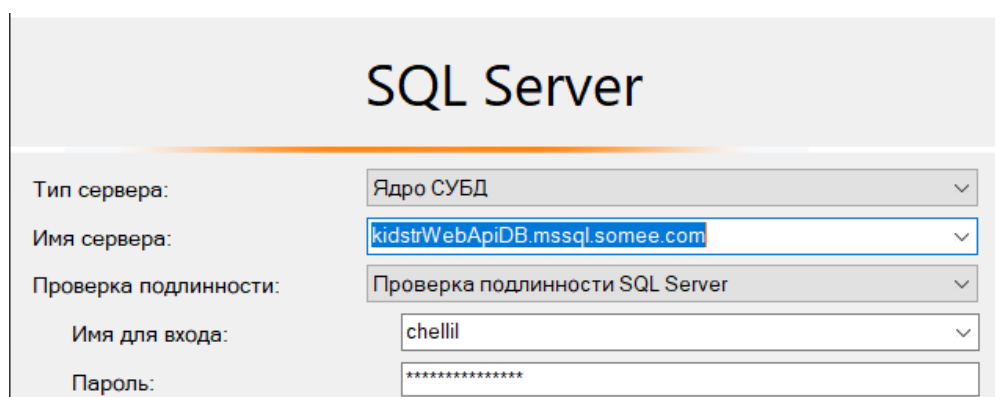


Рисунок 6 – Параметры доступа к серверу

В процессе реализации серверной части АИС было также написано веб-приложение ASP NET Core Web API под названием KidStrWebApi. Оно реализует заявленный при проектировании функционал Web API.

Взаимодействие API с БД выполняется с помощью Entity Framework Core [20], в соответствии с которым была создана модель данных на основе структуры БД (приложение В.2).

Для удобства модификации класс контроллера выполнен как partial и разделен на четыре файла в соответствии с выявленными при проектировании группами задач.

Представленный в приложении В.7 класс Logs содержит функции, вызываемые контроллером и добавляющие записи в конец файла logfile.txt в директории Log в корневой папке сервера Web API. Пример содержимого файла представлен на рисунке 7.

Созданный Web API был развернут на тестовом хостинге

<https://somee.com/> в соответствии с конфигурацией, указанной на рисунке 8.

Код модели, контекста контроллера и класса журналирования Web API представлен в приложении В.

```
5/30/2022 11:49:03 AM : User Manager log in
5/30/2022 11:49:14 AM : User Manager delete 1 row to AddServ
5/30/2022 11:49:21 AM : User Manager delete 1 row to AddServ
5/30/2022 11:49:36 AM : User Manager update 1 row to Service
5/30/2022 11:50:25 AM : User update 1 row to Ordr
5/30/2022 11:50:41 AM : User Admin log in
5/30/2022 11:51:08 AM : User Admin delete 1 row to Type
5/31/2022 9:44:56 PM : User Manager log in
5/31/2022 9:46:35 PM : User Admin log in
6/3/2022 12:53:53 PM : User Admin log in
```

Рисунок 7 – Текст журнала

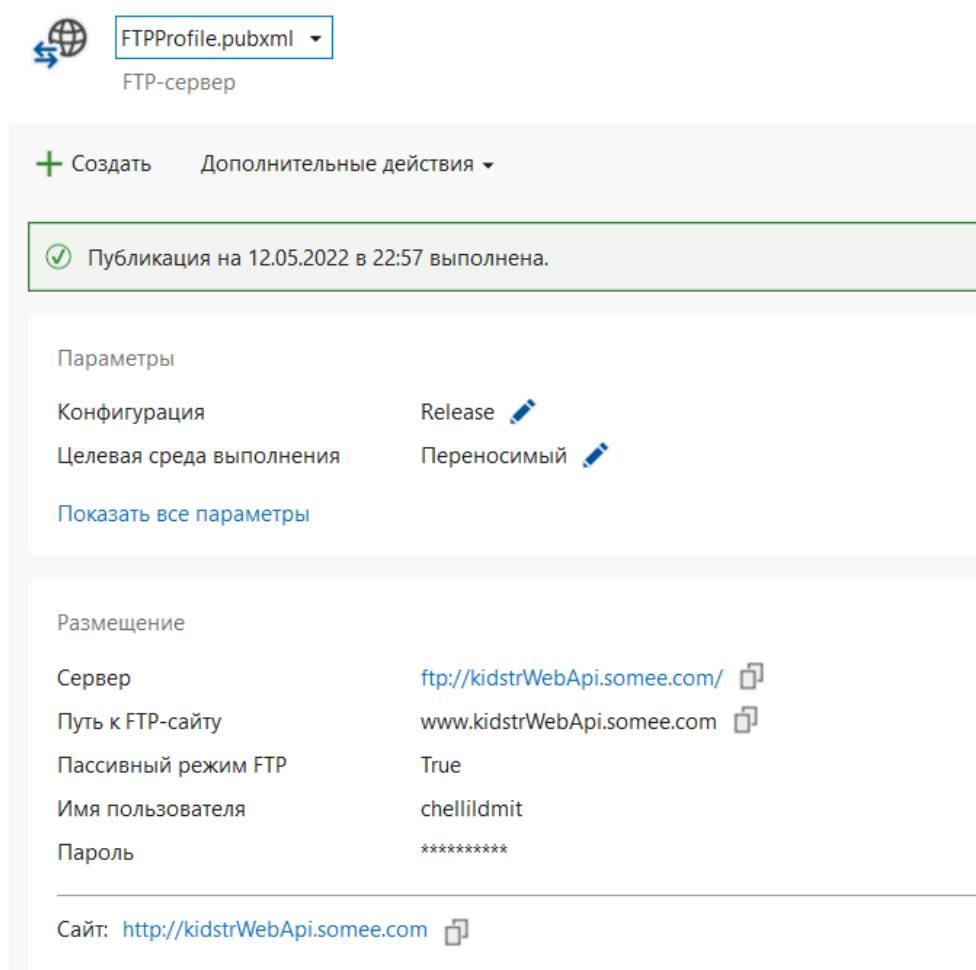


Рисунок 8 – Параметры публикации Web API

## 4.2 Разработка пользовательского интерфейса

Для разработки пользовательского интерфейса использовалась технология Windows Forms. [21]

В первую очередь, для возможности доступа к данным, для пользовательского интерфейса была частично воспроизведена модель данных, представленная в приложении Г.1. Также для непосредственного соединения с сервером и передачи данных в виде JSON-строк были созданы вспомогательные функции (приложения Г.2 и Г.4). Кроме того, для поддержания журналирования и многопользовательского режима в систему был добавлен статический класс текущего пользователя, в том числе хэширующий пароль при авторизации по алгоритму SHA256.

Получившаяся в итоге диаграмма классов представлена на рисунке 9.

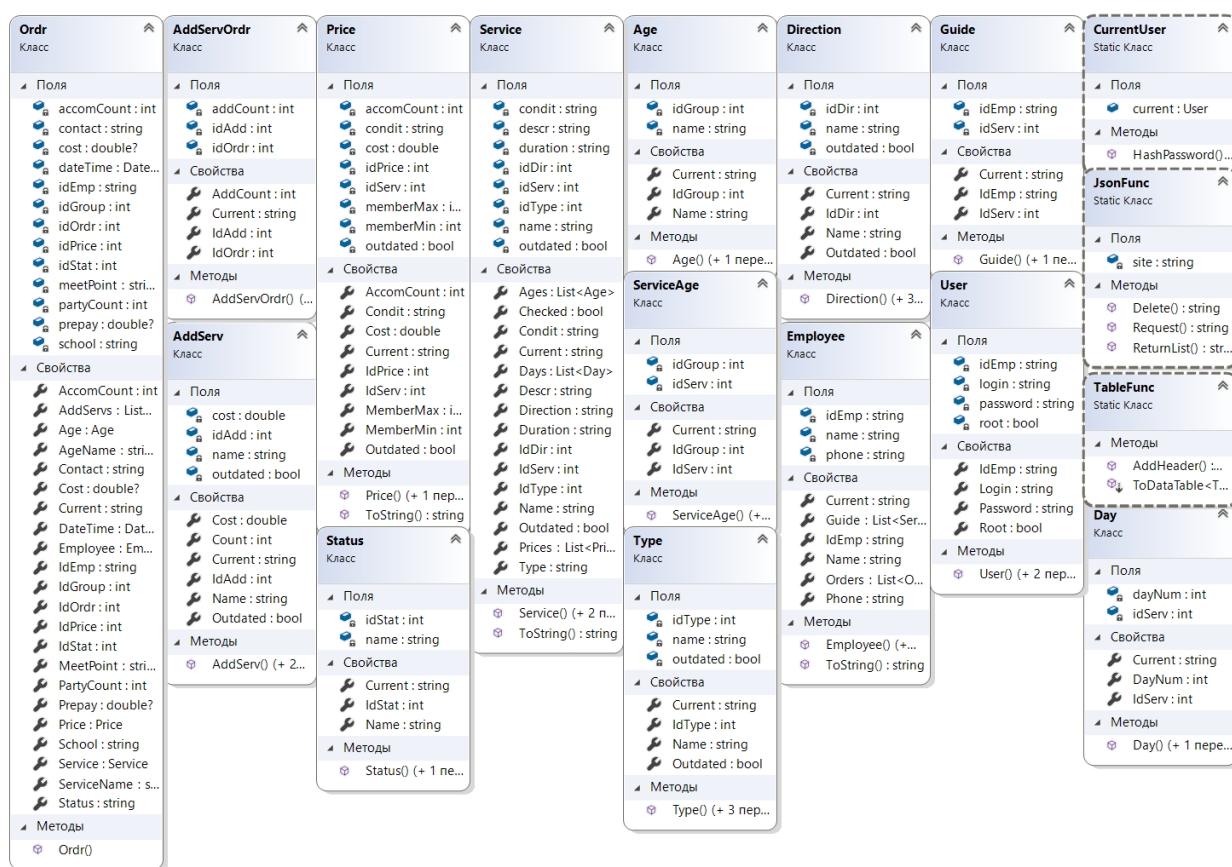


Рисунок 9 – Диаграмма классов модели для клиентской стороны

В процессе работы на основе иерархии экранных форм (рисунки 4 и 5) были созданы основные окна прототипа АИС с соблюдением общего дизайна. Для удобства работы размер шрифта всех элементов окон составляет 14пт, заголовков – 16пт.

При запуске приложения открывается окно авторизации, внешний вид которого представлен на рисунке 10. Пользователь обязан ввести логин и



пароль для доступа к системе. Символы пароля скрыты за «\*». Вход в систему происходит только при верно введенных данных. Пароль перед отправкой по сети хешируется.

В случае наличия прав администратора у пользователя и выбора режима «Войти как администратор» происходит вход на панель (главное окно) администратора. В ином случае на панель (главное окно) менеджера, однако если режим «Войти как администратор» выбран, а прав у пользователя недостаточно, вместо входа пользователю будет выдано предупреждение.

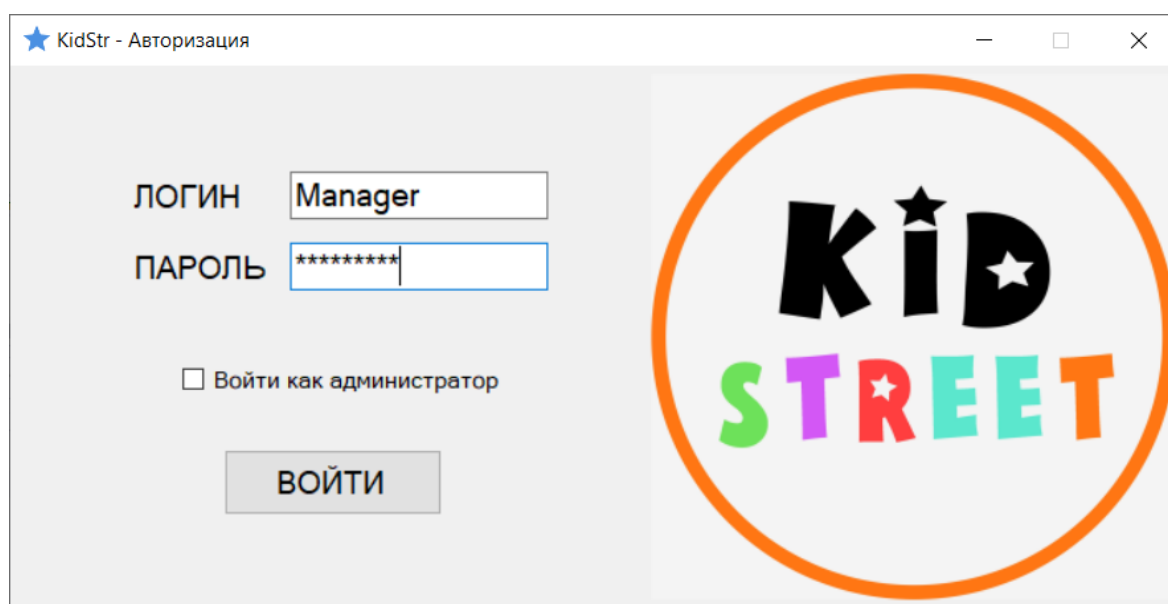


Рисунок 10 – Окно авторизации

Внешний вид главного окна пользователя на примере окна менеджера представлен на рисунке 11.

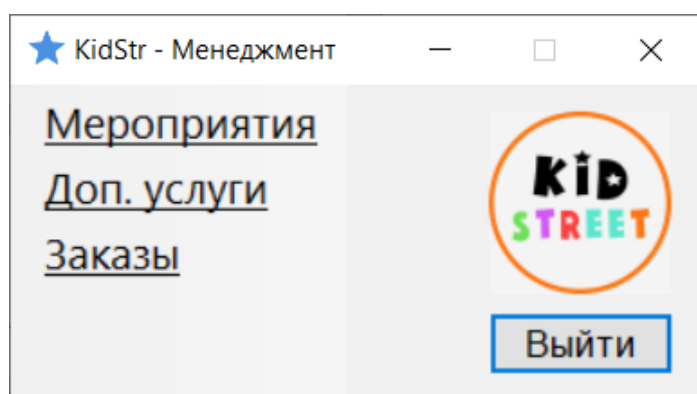


Рисунок 11 – Окно основного меню на примере главного окна менеджера

Внешний вид главного окна администратора и возможности выпадающего меню представлены на рисунке 12.

Из главных окон открываются окна списков данных, а также, в случае панели администратора, окно отчетов (статистик).

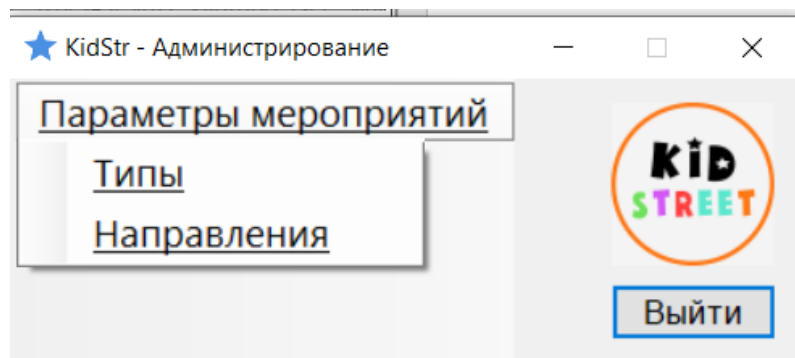


Рисунок 12 – Выпадающее меню в главном окне администратора

Список данных подразумевает наличие непосредственно таблицы с информацией, а также кнопок открытия окон добавления и изменения данных, окна детализированной информации при необходимости, а также кнопку удаления. Все кнопки помимо добавления заблокированы, если в таблице отсутствуют записи. Внешний вид списка данных на примере дополнительных услуг представлен на рисунке 13.

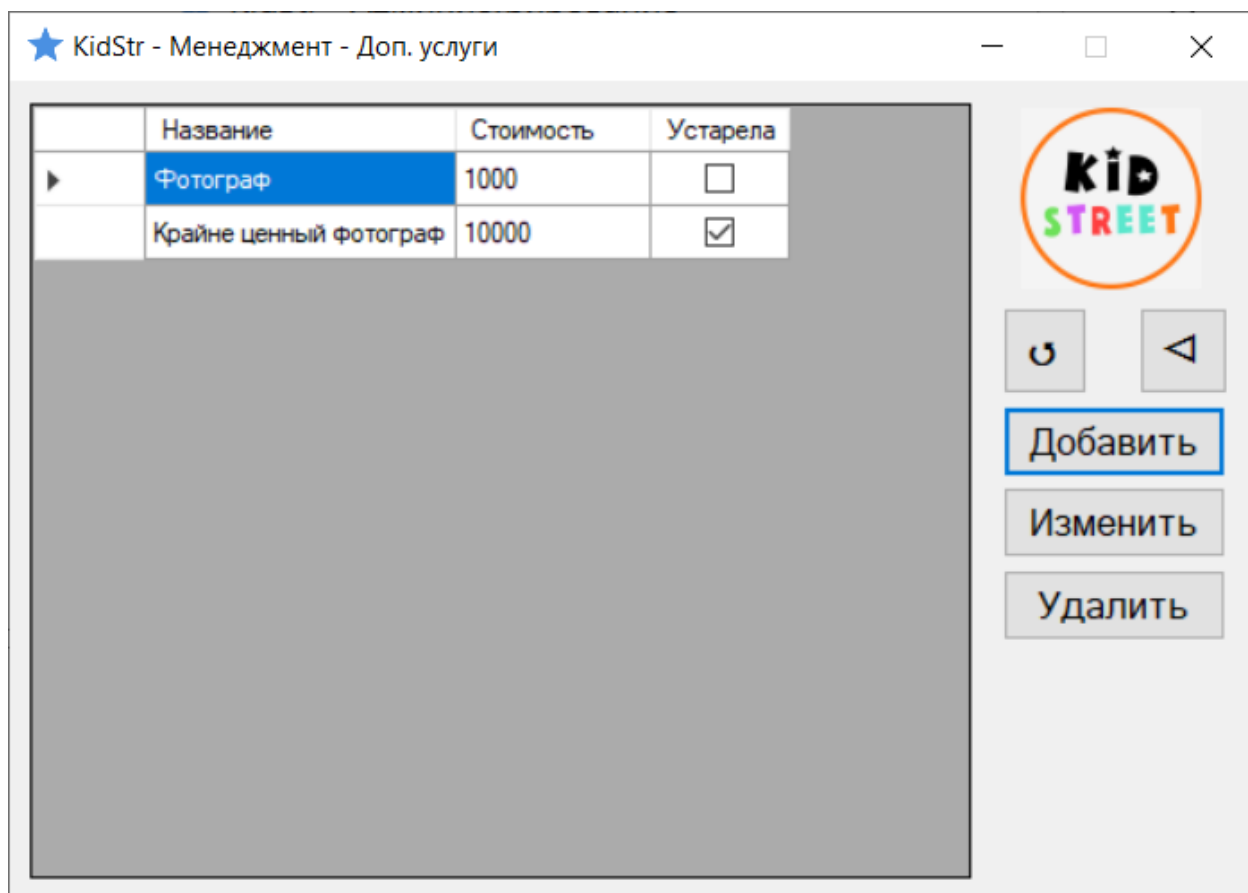


Рисунок 13 – Окно общего списка на примере списка дополнительных услуг

В связи с особенностями работы Entity Framework окна добавления и изменения данных работают практически аналогично и внутри кода представляют собой одно и то же окно с разными параметрами. Основное различие возникает при выборе метода контроллера. При вводе всех необходимых данных отправка их на сервер производится по кнопке сохранения. Окно детализации так же является тем же окном добавления и изменения, однако оно не позволяет менять выведенные данные и не отображает кнопку сохранения.

Окно добавления, изменения или детализации открывается в отдельном окне поверх окна списка. После его закрытия на «х» или после сохранения, в окне списка данных производится перезагрузка актуальной информации.

Внешний вид окна добавления данных на примере окна добавления данных сотрудника представлен на рисунке 14. Внешний вид окна изменения данных на примере изменения заказа представлен на рисунке 15.

ID	Название	Продолжительность	Описание	Ограничения	Устарела
1	Экскурсия1	2 часа 30 минут	Описание1	Ограничения1	<input type="checkbox"/>
2	Мастер-класс1	2 часа 30 минут	Описание1	Ограничения1	<input type="checkbox"/>

Рисунок 14 – Окно добавления данных на примере данных сотрудника

Основной задачей менеджеров является составление заказов на экскурсии и иные мероприятия. Окно добавления, изменения и детализации заказа на примере окна изменения представлено на рисунке 15. Стоимость заказа и предоплата пересчитываются в процессе работы при изменении

любых связанных с деньгами параметров («Вид цены» и «Доп. услуги»).

★ Изменить заказ № 2

Организация

Школа 1

Возраст/Класс

3 класс

Контактное лицо

Контакт 1

Дата и время встречи

26.05.2022 10:30

Точка встречи

Точка 2

Мероприятие

2 | Мастер-класс | 2 часа 30 м

Вид цены

6 | 11-15 чел., 1 сопр. | | 31500

Экскурсовод

3333 333333 | Name3 | +7

Доп. услуги\*:

	ID	Название	Стоимость	Устарела	Количество
▶	1	Фотограф	1000	<input type="checkbox"/>	3
	2	Другой фотограф	10000	<input checked="" type="checkbox"/>	0

Число

Участников

15

Сопровождающих

1

Стоимость: 34500

Предоплата: 11385

Статус: Оплачен

Сохранить

\*Для удаления доп. услуги из заказа удалите количество единиц предоставляемой услуги или замените их на 0.

Рисунок 15 – Окно изменения данных на примере заказа

На основе данных о заказах для модуля администратора по запросу генерируются отчеты в виде двух графиков – круговой диаграммы, описывающей соотношение возрастных групп в заказах, и столбчатой диаграммы, описывающей количество заказов за каждый месяц. Временной промежуток для отчетов можно указать дополнительно. По умолчанию учитываются заказы за все время работы АИС. Внешний вид окна отчетов представлен на рисунке 16.

Отчеты так же поддерживают выгрузку в файл с расширением **xlsx** (файл Microsoft Office Excel) обеих диаграмм, а также данных, на которых они строятся. Результат выгрузки представлен на рисунках 17 и 18 соответственно. Расположение и имя файла пользователь выбирает самостоятельно.

36

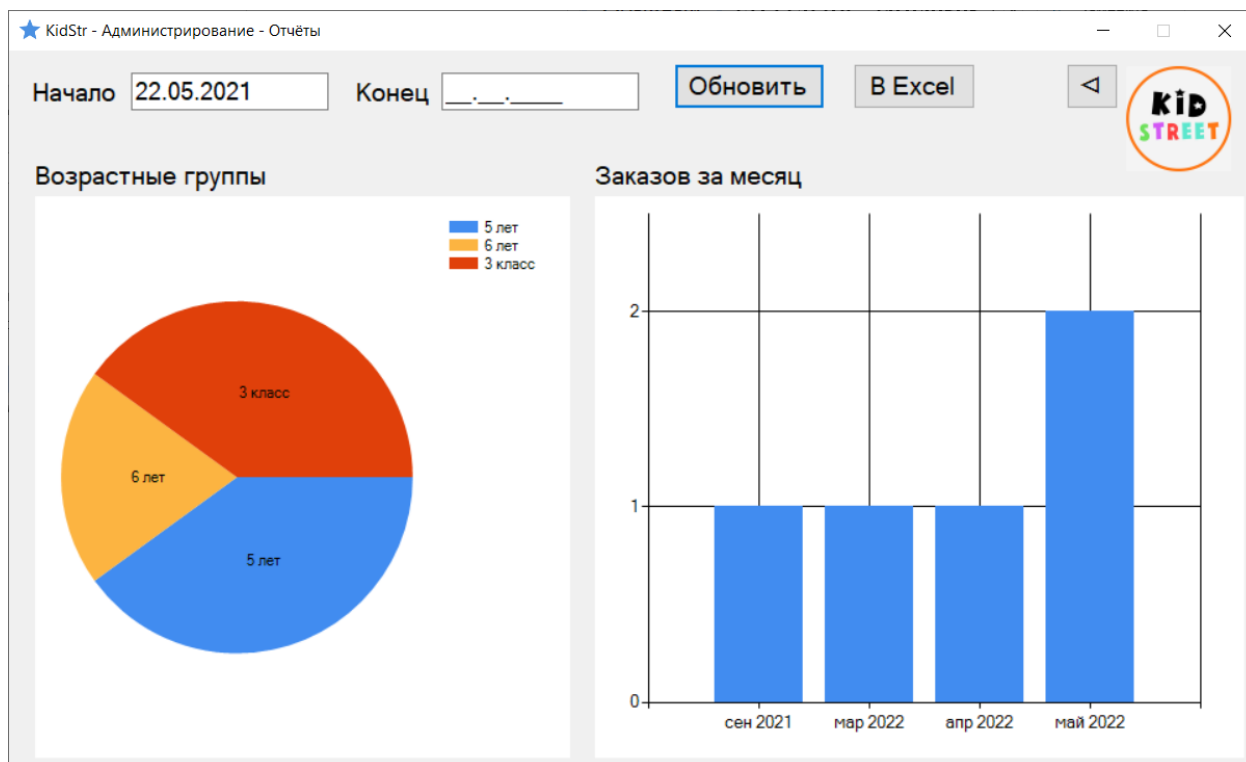


Рисунок 16 – Окно отображения отчетов

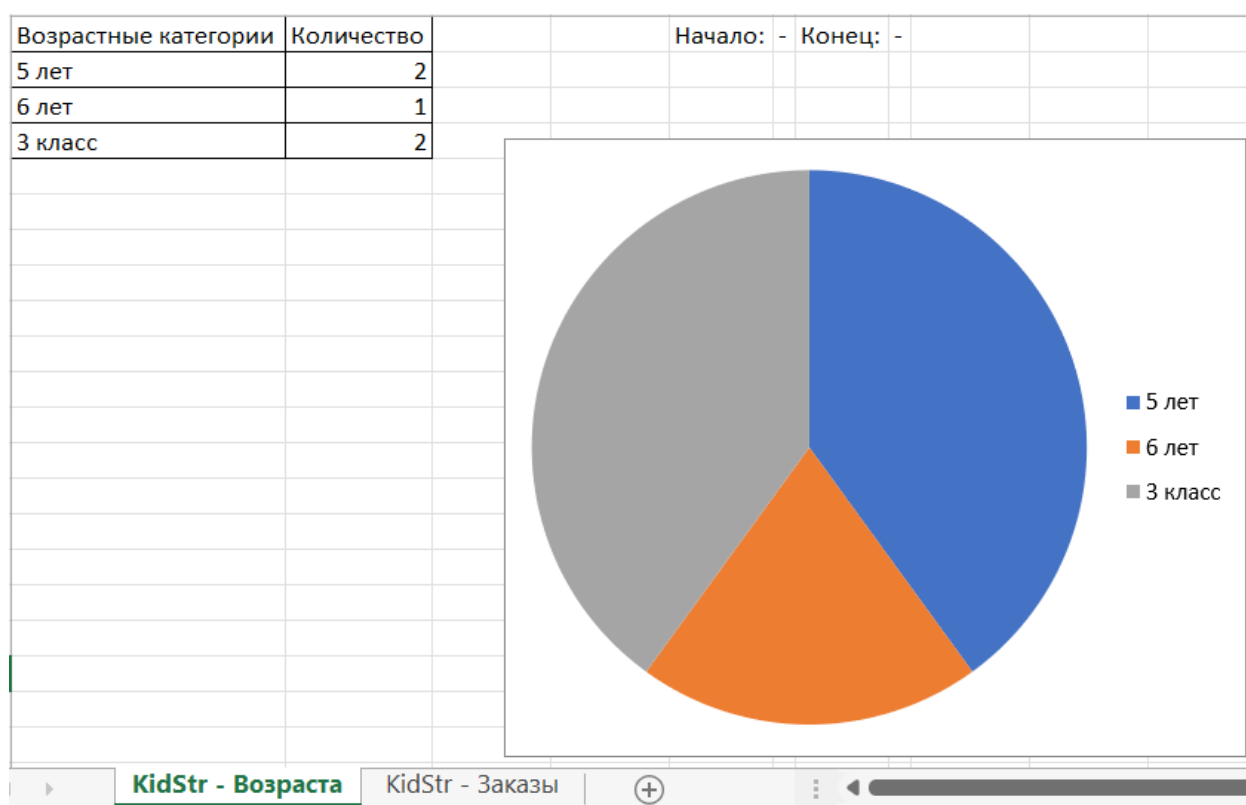


Рисунок 17 – Результат сохранения в файл первого отчета

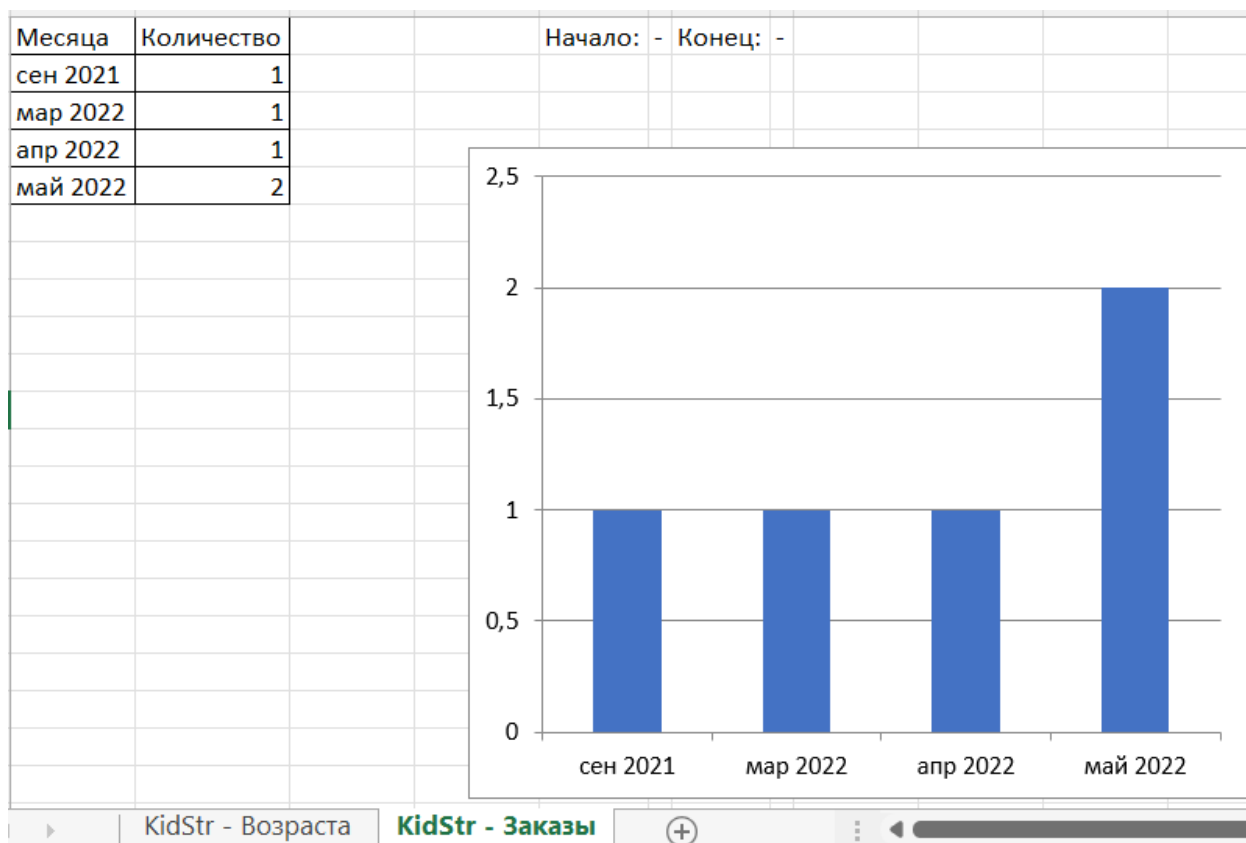


Рисунок 18 – Результат сохранения в файл второго отчета

Следует отметить, что нажатие на символ «х» во всех окнах кроме окон модификации и детализирования данных закрывает приложение. В связи с этим окна списка данных и отчетов содержат кнопки возврата на главное окно. Главные окна содержат кнопку выхода из системы. Также для поддержки актуальности данных окна списка данных содержат кнопку обновления данных.

Код окна авторизации, главного окна администратора, окна со списком дополнительных услуг и окна добавления, изменения и детализации данных о сотруднике представлен в приложении Г вместе с кодом необходимых вспомогательных классов функций. Код остальных форм опущен в связи с крайней схожестью с представленными примерами.

Для разработанного клиентского приложения был создан установочный пакет KidStrSetup средствами Setup Project Visual Studio, свойства которого представлены на рисунке 19. Текущая версия прототипа АИС считается версией 0.1.0. В качестве иконки приложения выбрано изображение звезды, представленное на рисунке 20 и взятое из открытых источников.

KidStrSetup Deployment Project Properties	
AddRemoveProgramsIcon	(None)
Author	Chelishcheva LD
BackwardCompatibleIDGeneration	False
Description	
DetectNewerInstalledVersion	True
InstallAllUsers	False
Keywords	
Localization	Russian
Manufacturer	SPbSTU
ManufacturerUrl	<a href="https://www.spbstu.ru/">https://www.spbstu.ru/</a>
PostBuildEvent	
PreBuildEvent	
ProductCode	{65485783-F3D5-49DF-9331-1CD248847D1C}
ProductName	KidStr
RemovePreviousVersions	True
RunPostBuildEvent	On successful build
SearchPath	
Subject	
SupportPhone	
SupportUrl	
TargetPlatform	x86
Title	Установка клиентского приложения KidStr
UpgradeCode	{62D8FEC1-B9BE-4702-8A8E-D4F2E5060507}
Version	0.1.0

Рисунок 19 – Свойства установочного пакета



Рисунок 20 – Иконка приложения

### 4.3 Тестирование АИС

Созданный проект был протестирован на соответствие заявленному функционалу и правильность выполнения поставленных задач. Результаты тестирования АИС представлены в приложении Д. Тестирование на предприятии с получением обратной связи от пользователей предполагается провести после завершения ВКР в рамках доработки прототипа.

## ЗАКЛЮЧЕНИЕ

В рамках выпускной квалификационной работы был разработан прототип АИС, позволяющий сократить работу менеджмента заказов на экскурсии в экскурсионно-образовательном центре КидСтрит. Полученный прототип сокращает расчеты, проводимые менеджерами, систематизирует используемую при этом информацию, а также позволяет работать из любого места, имея установленное клиентское приложение и зная свою комбинацию «логин – пароль».

Таким образом была достигнута поставленная цель: «Проектирование и разработка прототипа АИС управления заказами на экскурсии для экскурсионно-образовательного центра КидСтрит». В процессе выполнения были решены все заявленные задачи.

Несмотря на то, что полученный программный продукт является прототипом, он уже выполняет весь требуемый и заявленный функционал, а также прошел предварительное тестирование. В то же время, в рамках дальнейшего развития проекта планируется расширить и оптимизировать его функционал, включая:

- Добавление возможности сортировки и фильтрации при отображении списков данных.
- Расширение возможностей журналирования процессов, происходящих в АИС.
- Поддержка согласованности по чтению при работе нескольких пользователей над одними данными.
- Оптимизация дизайна пользовательского интерфейса и добавление дополнительных возможностей на основе обратной связи, полученной при планируемом тестировании на производстве.



## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Что такое CRM. – URL: [https://www.bitrix24.ru/articles/crm\\_what\\_is.php](https://www.bitrix24.ru/articles/crm_what_is.php) (дата обращения 15.02.2022).
2. Применяемый стек технологий. – URL: [http://rainbowsoft.ru/technology\\_stack](http://rainbowsoft.ru/technology_stack) (дата обращения 15.02.2022).
3. ХОСТИНГ - ЧТО ЭТО ТАКОЕ? КАКОЙ ХОСТИНГ ВЫБРАТЬ? – URL: <https://masterhost.ru/blog/hosting-chto-eto-takoe-kakoy-hosting-vyibrat/> (дата обращения 15.02.2022).
4. Журналирование. Почему нужны журналы и зачем их защищать? – URL: <https://acribia.ru/articles/journaling> (дата обращения 16.02.2022).
5. Экскурсии для школьников и студентов по Санкт–Петербургу и пригородам. Профориентация. – URL: <https://kid-street.ru/> (дата обращения 15.02.2022).
6. CRM 101: What is CRM? – URL: <https://www.salesforce.com/crm/what-is-crm/> (дата обращения 20.04.2022).
7. 3 types of CRM and how to choose the best one for your business. – URL: <https://www.zendesk.com/blog/3-types-crm-everything-need-know/> (дата обращения 20.04.2022).
8. САМО-тургид – программный комплекс для экскурсионных компаний. – URL: <https://samo.ru/excursion.html> (дата обращения 15.02.2022).
9. Автоматизация экскурсионного агентства. – URL: <https://bidbip.ru/excursions-providers-crm> (дата обращения 15.02.2022).
10. Комплексное решение для автоматизации экскурсионных агентств. – URL: <https://exaoffice.ru/applications/exaexcursions/> (дата обращения 15.02.2022).
11. Руденко А.М. Психология социально-культурного сервиса и туризма: учеб. пособие для студентов вузов / А.М. Руденко, М.А. Довгалева. – Ростов н/Д: Феникс, 2005. – 247 с.
12. Albahari J. C# 9.0 IN A NUTSHELL / J. Albahari. – Sebastopol : O'Reilly Media, Inc., 2021. – 1060 с.

13. Фреймворк Laravel. – URL: <https://unetway.com/blog/laravel-framework-review> (дата обращения 17.02.2022).
14. MS SQL. – URL: <https://bizzapps.ru/p/ms-sql/> (дата обращения 17.02.2022).
15. Справочное руководство по MySQL. – URL: <http://www.mysql.ru/docs/man/Features.html> (дата обращения 17.02.2022).
16. Шварц Б.А. MySQL. Оптимизация производительности / Б.А. Шварц, П.Н. Зайцев, В.Т. Ткаченко – М.: Наука, 2010. – 412 с.
17. Введение в JSON. – URL: <https://www.json.org/json-ru.html> (дата обращения 07.05.2022).
18. Руководство по Entity Framework Core. – URL: <https://metanit.com/sharp/entityframeworkcore/> (дата обращения 07.05.2022).
19. WEB API. – URL: <https://metanit.com/sharp/aspnet5/23.1.php> (дата обращения 07.05.2022).
20. Entity Framework Core 2 для ASP.NET Core MVC для профессионалов. : Пер. с англ. - СПб.: ООО "Диалектика", 2019. - 624 с.: ил.
21. NET Desktop Guide for Windows Forms. – URL: <https://docs.microsoft.com/en-us/dotnet/desktop/winforms/> (дата обращения 16.03.2021).

ПРИЛОЖЕНИЕ А

Диаграммы потоков данных АИС

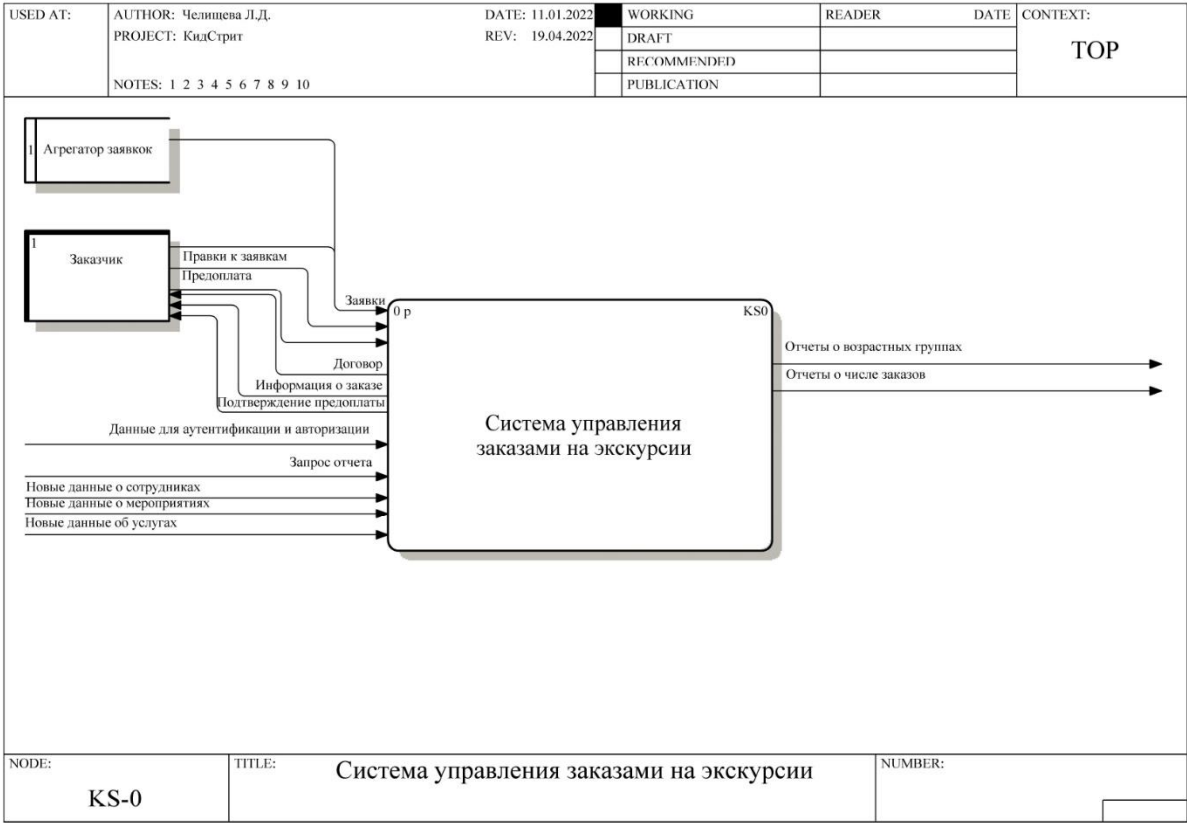


Рисунок А.1 – DFD уровня 0

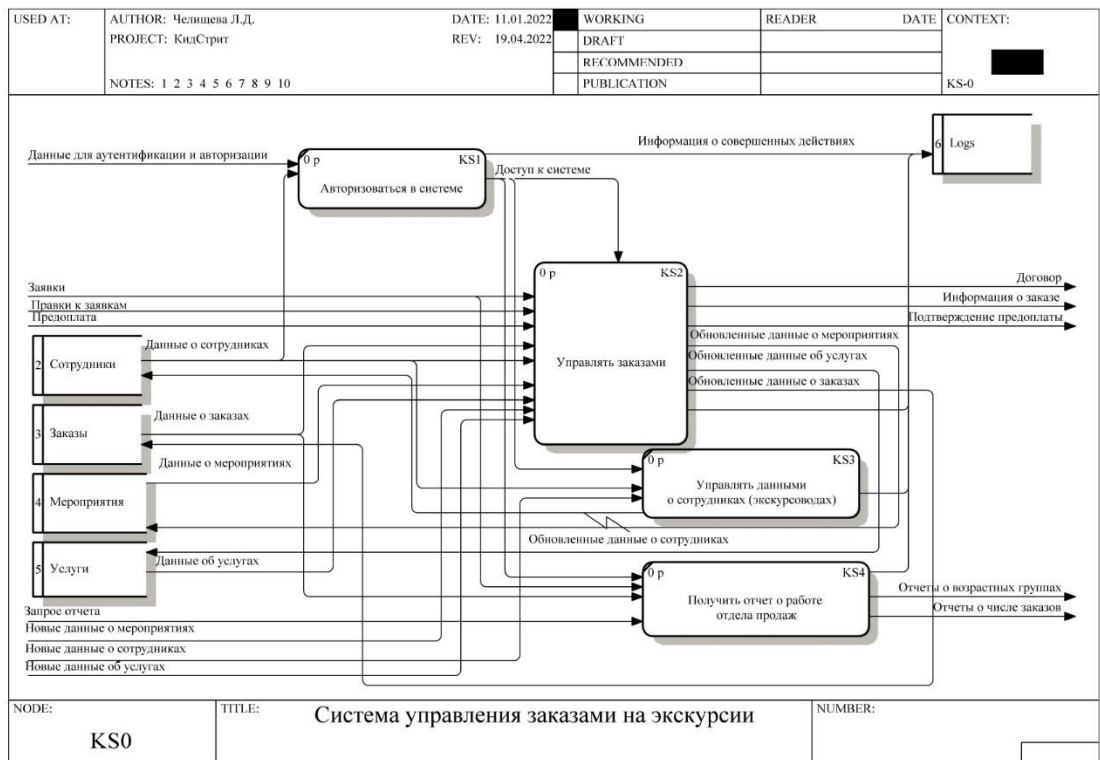


Рисунок А.2 – DFD уровня 1

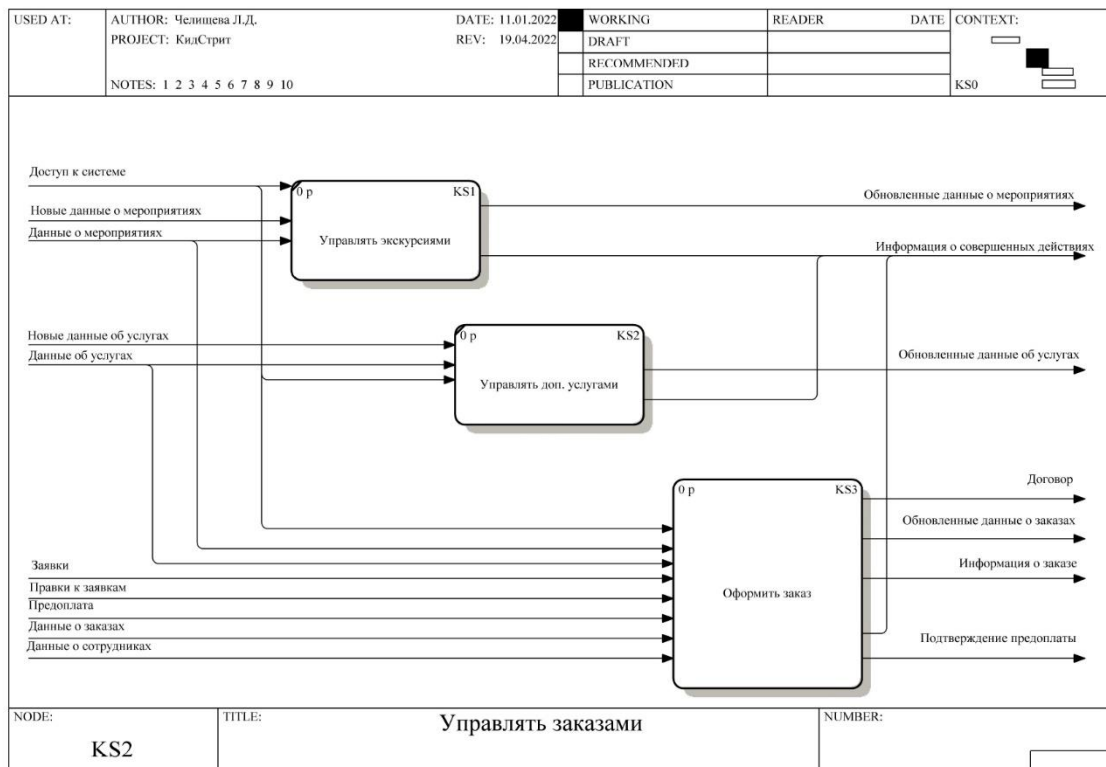


Рисунок А.3 – DFD уровня 2

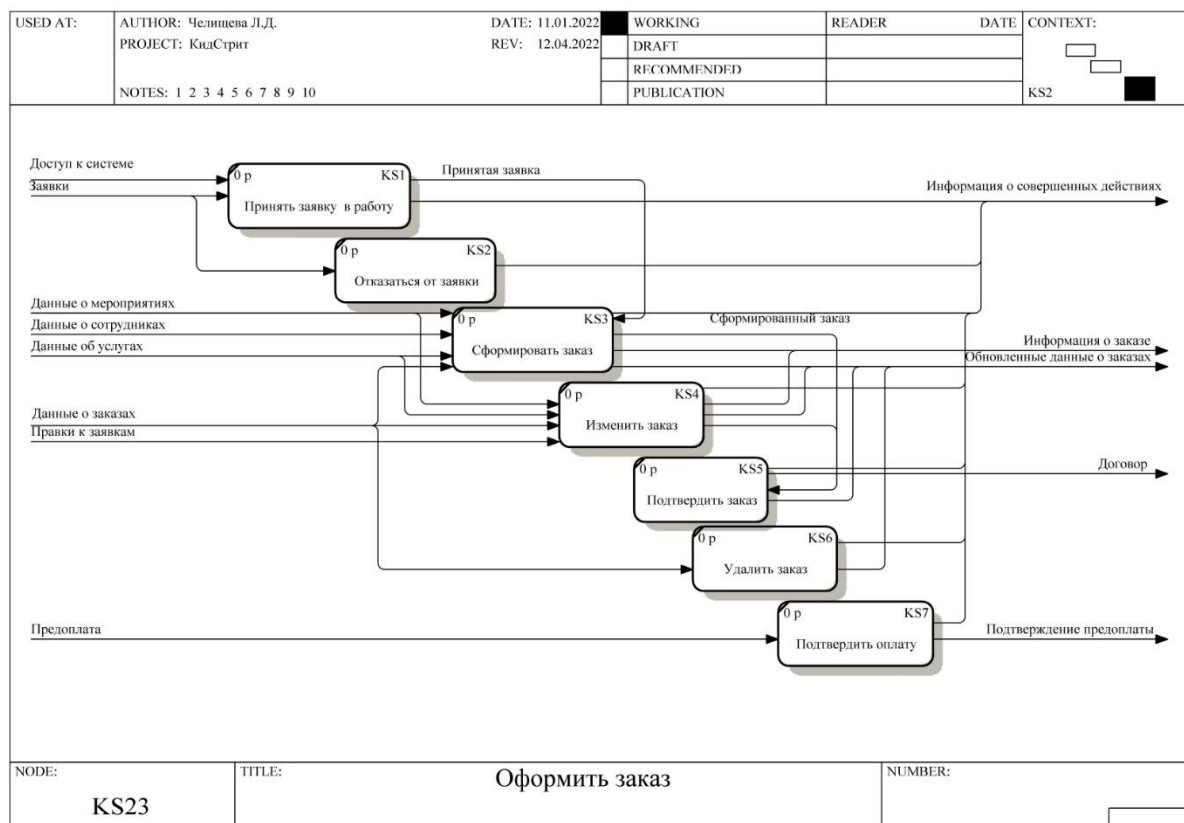


Рисунок А.4 – DFD уровня 3

## ПРИЛОЖЕНИЕ Б

### SQL-скрипт БД

```
CREATE TABLE [Employee]
(
    [id_emp]                nvarchar(15)    NOT NULL ,
    [name]                  nvarchar(50)    NOT NULL ,
    [phone]                 nvarchar(15)    NOT NULL ,
    PRIMARY KEY ([id_emp])
);

INSERT INTO [Employee] VALUES
('1111 111111', 'Name1', '+7(911)1111111'),
('2222 222222', 'Name2', '+7(922)2222222'),
('3333 333333', 'Name3', '+7(933)3333333');

CREATE TABLE [Users]
(
    [id_emp]                nvarchar(15)    NOT NULL ,
    [login]                 nvarchar(20)    UNIQUE NOT NULL ,
    [password]              nvarchar(100)   NOT NULL ,
    [root]                 bit    NOT NULL DEFAULT 0,
    PRIMARY KEY ([id_emp]),
    FOREIGN KEY ([id_emp]) REFERENCES [Employee]([id_emp])
        ON DELETE CASCADE
        ON UPDATE CASCADE
);

INSERT INTO [Users] VALUES
('1111 111111', 'Manager',
'dcccb657472b9f453acf0f7decdd5f8891d4eb997267de31b76bb02cd1bc4e1
7', 0), --Mpassword
('2222 222222', 'Admin',
'9fff15cdec1f0b4f804c2217cce94ad7194ebc18744892bc7bb52be9fd7bb94
2', 1); --Apassword

CREATE TABLE [Types]
(
    [id_type]              integer IDENTITY NOT NULL ,
    [name]                 nvarchar(20)    NOT NULL ,
    [outdated]            bit    NOT NULL DEFAULT 0,
    PRIMARY KEY ([id_type])
);

INSERT INTO [Types] ([name]) VALUES
(N'Экскурсия'),
(N'Мастер-класс'),
(N'Квест');

CREATE TABLE [Direction]
```

```

(
    [id_dir]                integer IDENTITY NOT NULL ,
    [name]                  nvarchar(20) NOT NULL ,
    [outdated]              bit NOT NULL DEFAULT 0,
    PRIMARY KEY ([id_dir])
);
INSERT INTO [Direction] ([name]) VALUES
(N'Наука и техника'),
(N'Производство'),
(N'Культура и искусство'),
(N'Мастер-классы'),
(N'В природу'),
(N'Интерактив'),
(N'По городу'),
(N'Музеи'),
(N'Пригороды'),
(N'Живой урок'),
(N'Новый год'),
(N'Выпускной'),
(N'Война и Блокада'),
(N'Туры в Петербург');

CREATE TABLE [Service]
(
    [id_serv]                integer IDENTITY NOT NULL ,
    [name]                  nvarchar(50) NOT NULL ,
    [duration]              nvarchar(20) NOT NULL ,
    [descr]                 nvarchar(max) NULL ,
    [condit]                nvarchar(max) NULL ,
    [id_type]               integer NOT NULL ,
    [id_dir]                integer NOT NULL ,
    [outdated]              bit NOT NULL DEFAULT 0,
    PRIMARY KEY ([id_serv]),
    FOREIGN KEY ([id_type]) REFERENCES [Types]([id_type])
        ON DELETE NO ACTION
        ON UPDATE CASCADE,
    FOREIGN KEY ([id_dir]) REFERENCES [Direction]([id_dir])
        ON DELETE NO ACTION
        ON UPDATE CASCADE
);
INSERT INTO [Service]
([name],[duration],[descr],[condit],[id_type],[id_dir]) VALUES
(N'Экскурсия1', N'2 часа 30 минут', N'Описание1', N'Ограничения1',
1, 2),
(N'Мастер-класс1', N'2 часа 30 минут', N'Описание1',
N'Ограничения1', 2, 2);

CREATE TABLE [Age]
(
    [id_group]              integer IDENTITY NOT NULL ,
    [name]                  nvarchar(15) NOT NULL ,
    PRIMARY KEY ([id_group])
);

```

```

INSERT INTO [Age] ([name]) VALUES
(N'5 лет'),
(N'6 лет'),
(N'1 класс'),
(N'2 класс'),
(N'3 класс'),
(N'4 класс'),
(N'5 класс'),
(N'6 класс'),
(N'7 класс'),
(N'8 класс'),
(N'9 класс'),
(N'10 класс'),
(N'11 класс'),
(N'Студенты'),
(N'Взрослые'),
(N'Смешан. ');

CREATE TABLE [Service_Age]
(
    [id_serv]            integer NOT NULL ,
    [id_group]           integer NOT NULL ,
    PRIMARY KEY ([id_serv],[id_group]),
    FOREIGN KEY ([id_serv]) REFERENCES [Service]([id_serv])
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    FOREIGN KEY ([id_group]) REFERENCES [Age]([id_group])
        ON DELETE CASCADE
        ON UPDATE CASCADE
);

INSERT INTO [Service_Age] VALUES
(1, 1),
(1, 2),
(1, 3),
(2, 4),
(2, 5);

CREATE TABLE [Guide]
(
    [id_emp]             nvarchar(15) NOT NULL ,
    [id_serv]            integer NOT NULL ,
    PRIMARY KEY ([id_emp],[id_serv]),
    FOREIGN KEY ([id_emp]) REFERENCES [Employee]([id_emp])
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    FOREIGN KEY ([id_serv]) REFERENCES [Service]([id_serv])
        ON DELETE CASCADE
        ON UPDATE CASCADE
);

INSERT INTO [Guide] VALUES
('3333 333333', 1),
('3333 333333', 2);

```



```

CREATE TABLE [Days]
(
    [id_serv]            integer NOT NULL ,
    [day_num]            integer NOT NULL ,
    PRIMARY KEY ([id_serv],[day_num]),
    FOREIGN KEY ([id_serv]) REFERENCES [Service]([id_serv])
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
INSERT INTO [Days] VALUES
(1, 1),
(1, 2),
(1, 3),
(2, 4),
(2, 5);

CREATE TABLE [AddServ]
(
    [id_add]             integer IDENTITY NOT NULL ,
    [name]               nvarchar(50) NOT NULL ,
    [cost]               float NOT NULL DEFAULT 0,
    [outdated]          bit NOT NULL DEFAULT 0,
    PRIMARY KEY ([id_add])
);
INSERT INTO [AddServ] ([name], [cost]) VALUES
(N'Фотораф', 1000);

CREATE TABLE [Price]
(
    [id_price]           integer IDENTITY NOT NULL ,
    [id_serv]            integer NOT NULL ,
    [member_min]         integer NOT NULL DEFAULT 0,
    [member_max]         integer NULL ,
    [accom_count]        integer NOT NULL DEFAULT 0,
    [condit]             nvarchar(max) ,
    [cost]               float NOT NULL DEFAULT 0,
    [outdated]          bit NOT NULL DEFAULT 0,
    PRIMARY KEY ([id_price]),
    FOREIGN KEY ([id_serv]) REFERENCES [Service]([id_serv])
        ON DELETE CASCADE
        ON UPDATE CASCADE
);

INSERT INTO [Price] ([id_serv], [member_min], [member_max],
[accom_count], [cost]) VALUES
(1, 11, 15, 1, 31500),
(1, 16, 20, 2, 38500),
(1, 21, 30, 2, 40000),
(1, 31, 40, 2, 41500),
(1, 41, 45, 2, 43000),
(2, 11, 15, 1, 31500),
(2, 16, 20, 2, 38500),
(2, 21, 30, 2, 40000);

```

```

CREATE TABLE [Status]
(
    [id_stat]                integer IDENTITY NOT NULL ,
    [name]                   nvarchar(10) NOT NULL ,
    PRIMARY KEY ([id_stat])
);
INSERT INTO [Status] ([name]) VALUES
(N'Оформлен'),
(N'Оплачен'),
(N'Выполнен'),
(N'Отменен');

CREATE TABLE [Ordr]
(
    [id_ordr]                integer IDENTITY NOT NULL ,
    [school]                 nvarchar(50) NOT NULL ,
    [id_group]               integer NOT NULL ,
    [party_count]            integer NOT NULL ,
    [accom_count]            integer NOT NULL DEFAULT 0,
    [contact]                nvarchar(50) NOT NULL ,
    [date_time]              datetime NOT NULL ,
    [meet_point]             nvarchar(20) NOT NULL ,
    [cost]                   float NULL ,
    [prepay]                 float NULL ,
    [id_price]               integer NOT NULL ,
    [id_stat]                integer NOT NULL ,
    [id_emp]                 nvarchar(15) NULL ,
    PRIMARY KEY ([id_ordr]),
    FOREIGN KEY ([id_price]) REFERENCES [Price]([id_price])
        ON DELETE NO ACTION
        ON UPDATE CASCADE,
    FOREIGN KEY ([id_group]) REFERENCES [Age]([id_group])
        ON DELETE NO ACTION
        ON UPDATE CASCADE,
    FOREIGN KEY ([id_stat]) REFERENCES [Status]([id_stat])
        ON DELETE NO ACTION
        ON UPDATE CASCADE,
    FOREIGN KEY ([id_emp]) REFERENCES [Employee]([id_emp])
        ON DELETE SET NULL
        ON UPDATE CASCADE
);

CREATE TABLE [AddServ_Ordr]
(
    [id_add]                 integer NOT NULL ,
    [id_ordr]                integer NOT NULL ,
    [add_count]              integer NOT NULL DEFAULT 1,
    PRIMARY KEY ([id_add],[id_ordr]),
    FOREIGN KEY ([id_add]) REFERENCES [AddServ]([id_add])
        ON DELETE NO ACTION
        ON UPDATE CASCADE,

```

```
FOREIGN KEY ([id_ordr]) REFERENCES [Ordr]([id_ordr])  
    ON DELETE CASCADE  
    ON UPDATE CASCADE  
);
```

## ПРИЛОЖЕНИЕ В

### Код основных элементов Web API

#### В.1 Контекст

```
public class KidstrContext : DbContext
{
    public DbSet<AddServ> AddServs { get; set; }
    public DbSet<AddServOrdr> AddServOrdrs { get; set; }
    public DbSet<Age> Ages { get; set; }
    public DbSet<Day> Days { get; set; }
    public DbSet<Direction> Directions { get; set; }
    public DbSet<Employee> Employees { get; set; }
    public DbSet<Guide> Guides { get; set; }
    public DbSet<Ordr> Ordrs { get; set; }
    public DbSet<Price> Prices { get; set; }
    public DbSet<Service> Services { get; set; }
    public DbSet<ServiceAge> ServiceAges { get; set; }
    public DbSet<Status> Statuses { get; set; }
    public DbSet<Models.Type> Types { get; set; }
    public DbSet<User> Users { get; set; }

    public KidstrContext(DbContextOptions<KidstrContext>
options)
        : base(options)
    {
        Database.EnsureCreated();
    }

    protected override void OnModelCreating(ModelBuilder
modelBuilder)
    {
        modelBuilder.Entity<AddServ>(entity =>
        {
            entity.HasKey(e => e.IdAdd);
            entity.Property(e => e.IdAdd).ValueGeneratedOnAdd();
        });

        modelBuilder.Entity<AddServOrdr>(entity =>
        {
            entity.HasKey(e => new { e.IdAdd, e.IdOrdr });
        });

        modelBuilder.Entity<Age>(entity =>
        {
            entity.HasKey(e => e.IdGroup);
        });
    }
}
```

```

        entity.Property(e
e.IdGroup).ValueGeneratedOnAdd();
    });

    modelBuilder.Entity<Day>(entity =>
    {
        entity.HasKey(e => new { e.IdServ, e.DayNum });
    });

    modelBuilder.Entity<Direction>(entity =>
    {
        entity.HasKey(e => e.IdDir);
        entity.Property(e
e.IdDir).ValueGeneratedOnAdd();
    });

    modelBuilder.Entity<Employee>(entity =>
    {
        entity.HasKey(e => e.IdEmp);
    });

    modelBuilder.Entity<Guide>(entity =>
    {
        entity.HasKey(e => new { e.IdEmp, e.IdServ });
    });

    modelBuilder.Entity<Ordr>(entity =>
    {
        entity.HasKey(e => e.IdOrdr);
        entity.Property(e
e.IdOrdr).ValueGeneratedOnAdd();
    });

    modelBuilder.Entity<Price>(entity =>
    {
        entity.HasKey(e => e.IdPrice);
        entity.Property(e
e.IdPrice).ValueGeneratedOnAdd();
    });

    modelBuilder.Entity<Service>(entity =>
    {
        entity.HasKey(e => e.IdServ);
        entity.Property(e
e.IdServ).ValueGeneratedOnAdd();
    });

    modelBuilder.Entity<ServiceAge>(entity =>
    {
        entity.HasKey(e => new { e.IdServ, e.IdGroup });
    });

    modelBuilder.Entity<Status>(entity =>

```

```

        {
            entity.HasKey(e => e.IdStat);
            entity.Property(e
e.IdStat).ValueGeneratedOnAdd();
        });

        modelBuilder.Entity<Models.Type>(entity =>
        {
            entity.HasKey(e => e.IdType);
            entity.Property(e
e.IdType).ValueGeneratedOnAdd();
        });

        modelBuilder.Entity<User>(entity =>
        {
            entity.HasKey(e => e.IdEmp);
        });
    }
}

```

## B.2 Код модели со стороны Web API

```

[Table("AddServ")]
public class AddServ
{
    private int idAdd;
    private string name;
    private double cost;
    private bool outdated;

    public AddServ()
    {
    }

    public AddServ(string name, double cost, bool outdated)
    {
        this.name = name;
        this.cost = cost;
        this.outdated = outdated;
    }

    public AddServ(string name, double cost)
    {
        this.name = name;
        this.cost = cost;
    }

    [Key]
    [Column("id_add")]
    [JsonProperty("IdAdd")]
    public int IdAdd { get => idAdd; set => idAdd = value; }
    [Column("name")]
    [JsonProperty("Name")]

```

```

    public string Name { get => name; set => name = value; }
    [Column("cost")]
    [DefaultValue(0)]
    [JsonProperty("Cost")]
    public double Cost { get => cost; set => cost = value; }
    [Column("outdated")]
    [DefaultValue(false)]
    [JsonProperty("Outdated")]
    public bool Outdated { get => outdated; set => outdated =
value; }
    [NotMapped]
    [JsonProperty("Current")]
    public string Current { get; set; }
}

[Table("AddServ_Ordre")]
public class AddServOrdre
{
    private int idAdd;
    private int idOrdre;
    private int addCount;

    public AddServOrdre()
    {
    }

    public AddServOrdre(int idAdd, int idOrdre, int addCount)
    {
        this.idAdd = idAdd;
        this.idOrdre = idOrdre;
        this.addCount = addCount;
    }

    public AddServOrdre(int idAdd, int idOrdre)
    {
        this.idAdd = idAdd;
        this.idOrdre = idOrdre;
    }

    [Key]
    [Column("id_add")]
    [JsonProperty("IdAdd")]
    public int IdAdd { get => idAdd; set => idAdd = value; }
    [Key]
    [Column("id_ordre")]
    [JsonProperty("IdOrdre")]
    public int IdOrdre { get => idOrdre; set => idOrdre = value; }
    [Column("add_count")]
    [DefaultValue(1)]
    [JsonProperty("AddCount")]
    public int AddCount { get => addCount; set => addCount =
value; }
    [NotMapped]
    [JsonProperty("Current")]

```

```

        public string Current { get; set; }
    }

    [Table("Age")]
    public class Age
    {
        private int idGroup;
        private string name;

        public Age()
        {
        }

        public Age(string name)
        {
            this.name = name;
        }

        [Key]
        [Column("id_group")]
        [JsonProperty("IdGroup")]
        public int IdGroup { get => idGroup; set => idGroup = value; }

        [Column("name")]
        [JsonProperty("Name")]
        public string Name { get => name; set => name = value; }
        [NotMapped]
        [JsonProperty("Current")]
        public string Current { get; set; }
    }

    [Table("Days")]
    public class Day
    {
        private int idServ;
        private int dayNum;

        public Day()
        {
        }

        public Day(int idServ, int dayNum)
        {
            this.idServ = idServ;
            this.dayNum = dayNum;
        }

        [Key]
        [Column("id_serv")]
        [JsonProperty("IdServ")]
        public int IdServ { get => idServ; set => idServ = value; }
        [Key]
        [Column("day_num")]

```



```

    [JsonProperty("DayNum")]
    public int DayNum { get => dayNum; set => dayNum = value; }
    [NotMapped]
    [JsonProperty("Current")]
    public string Current { get; set; }
}

[Table("Direction")]
public class Direction
{
    private int idDir;
    private string name;
    private bool outdated;

    public Direction()
    {
    }

    public Direction(string name, bool outdated)
    {
        this.name = name;
        this.outdated = outdated;
    }

    public Direction(string name)
    {
        this.name = name;
    }

    [Key]
    [Column("id_dir")]
    [JsonProperty("IdDir")]
    public int IdDir { get => idDir; set => idDir = value; }
    [Column("name")]
    [JsonProperty("Name")]
    public string Name { get => name; set => name = value; }
    [Column("outdated")]
    [DefaultValue(false)]
    [JsonProperty("Outdated")]
    public bool Outdated { get => outdated; set => outdated =
value; }
    [NotMapped]
    [JsonProperty("Current")]
    public string Current { get; set; }
}

[Table("Employee")]
public class Employee
{
    private string idEmp;
    private string name;
    private string phone;

```

```

public Employee()
{
}

public Employee(string idEmp, string name, string phone)
{
    this.idEmp = idEmp;
    this.name = name;
    this.phone = phone;
}

[Key]
[Column("id_emp")]
[JsonProperty("IdEmp")]
public string IdEmp { get => idEmp; set => idEmp = value; }
[Column("name")]
[JsonProperty("Name")]
public string Name { get => name; set => name = value; }
[Column("phone")]
[JsonProperty("Phone")]
public string Phone { get => phone; set => phone = value; }
[NotMapped]
[JsonProperty("Current")]
public string Current { get; set; }
}

[Table("Guide")]
public class Guide
{
    private string idEmp;
    private int idServ;

    public Guide()
    {
    }

    public Guide(string idEmp, int idServ)
    {
        this.idEmp = idEmp;
        this.idServ = idServ;
    }

    [Key]
    [Column("id_emp")]
    [JsonProperty("IdEmp")]
    public string IdEmp { get => idEmp; set => idEmp = value; }
    [Key]
    [Column("id_serv")]
    [JsonProperty("IdServ")]
    public int IdServ { get => idServ; set => idServ = value; }
    [NotMapped]
    [JsonProperty("Current")]
    public string Current { get; set; }
}

```

```

}

[Table("Ordr")]
public class Ordr
{
    private int idOrdr;
    private string school;
    private int idGroup;
    private int partyCount;
    private int accomCount;
    private string contact;
    private DateTime dateTime;
    private string meetPoint;
    private double? cost;
    private double? prepay;
    private int idPrice;
    private int idStat;
    private string idEmp;

    public Ordr()
    {
    }

    [Key]
    [Column("id_ordr")]
    [JsonProperty("IdOrdr")]
    public int IdOrdr { get => idOrdr; set => idOrdr = value; }
    [Column("school")]
    [JsonProperty("School")]
    public string School { get => school; set => school = value;
}

    [Column("id_group")]
    [JsonProperty("IdGroup")]
    public int IdGroup { get => idGroup; set => idGroup = value;
}

    [Column("party_count")]
    [JsonProperty("PartyCount")]
    public int PartyCount { get => partyCount; set => partyCount
= value; }
    [Column("accom_count")]
    [DefaultValue(0)]
    [JsonProperty("AccomCount")]
    public int AccomCount { get => accomCount; set => accomCount
= value; }
    [Column("contact")]
    [JsonProperty("Contact")]
    public string Contact { get => contact; set => contact =
value; }
    [Column("date_time", TypeName = "datetime")]
    [JsonProperty("DateTime")]
    public DateTime DateTime { get => dateTime; set => dateTime
= value; }
    [Column("meet_point")]

```

```

        [JsonProperty("MeetPoint")]
        public string MeetPoint { get => meetPoint; set => meetPoint
= value; }
        [Column("cost")]
        [JsonProperty("Cost")]
        public double? Cost { get => cost; set => cost = value; }
        [Column("prepay")]
        [JsonProperty("Prepay")]
        public double? Prepay { get => prepay; set => prepay = value;
}

        [Column("id_price")]
        [JsonProperty("IdPrice")]
        public int IdPrice { get => idPrice; set => idPrice = value;
}

        [Column("id_stat")]
        [JsonProperty("IdStat")]
        public int IdStat { get => idStat; set => idStat = value; }
        [Column("id_emp")]
        [JsonProperty("IdEmp")]
        public string IdEmp { get => idEmp; set => idEmp = value; }
        [NotMapped]
        [JsonProperty("Current")]
        public string Current { get; set; }
}

[Table("Price")]
public class Price
{
    private int idPrice;
    private int idServ;
    private int memberMin;
    private int? memberMax;
    private int accomCount;
    private string condit;
    private double cost;
    private bool outdated;

    public Price()
    {
    }

    public Price(int idServ, int memberMin, int? memberMax, int
accomCount, string condit, double cost, bool outdated)
    {
        this.idServ = idServ;
        this.memberMin = memberMin;
        this.memberMax = memberMax;
        this.accomCount = accomCount;
        this.condit = condit;
        this.cost = cost;
        this.outdated = outdated;
    }
}

```

```

        [Key]
        [Column("id_price")]
        [JsonProperty("IdPrice")]
        public int IdPrice { get => idPrice; set => idPrice = value;
    }

    [Column("id_serv")]
    [JsonProperty("IdServ")]
    public int IdServ { get => idServ; set => idServ = value; }
    [Column("member_min")]
    [DefaultValue(0)]
    [JsonProperty("MemberMin")]
    public int MemberMin { get => memberMin; set => memberMin =
value; }
    [Column("member_max")]
    [JsonProperty("MemberMax")]
    public int? MemberMax { get => memberMax; set => memberMax =
value; }
    [Column("accom_count")]
    [DefaultValue(0)]
    [JsonProperty("AccomCount")]
    public int AccomCount { get => accomCount; set => accomCount
= value; }
    [Column("condit")]
    [JsonProperty("Condit")]
    public string Condit { get => condit; set => condit = value;
}

    [Column("cost")]
    [DefaultValue(0)]
    [JsonProperty("Cost")]
    public double Cost { get => cost; set => cost = value; }
    [Column("outdated")]
    [DefaultValue(false)]
    [JsonProperty("Outdated")]
    public bool Outdated { get => outdated; set => outdated =
value; }
    [NotMapped]
    [JsonProperty("Current")]
    public string Current { get; set; }
}

[Table("Service")]
public class Service
{
    private int idServ;
    private string name;
    private string duration;
    private string descr;
    private string condit;
    private int idType;
    private int idDir;
    private bool outdated;

    public Service()

```

```

    {
    }

    public Service(string name, string duration, string descr,
string condit, int idType, int idDir)
    {
        this.name = name;
        this.duration = duration;
        this.descr = descr;
        this.condit = condit;
        this.idType = idType;
        this.idDir = idDir;
    }

    public Service(string name, string duration, string descr,
string condit, int idType, int idDir, bool outdated)
    {
        this.name = name;
        this.duration = duration;
        this.descr = descr;
        this.condit = condit;
        this.idType = idType;
        this.idDir = idDir;
        this.outdated = outdated;
    }

    [Key]
    [Column("id_serv")]
    [JsonProperty("IdServ")]
    public int IdServ { get => idServ; set => idServ = value; }
    [Column("name")]
    [JsonProperty("Name")]
    public string Name { get => name; set => name = value; }
    [Column("duration")]
    [JsonProperty("Duration")]
    public string Duration { get => duration; set => duration =
value; }
    [Column("descr")]
    [JsonProperty("Descr")]
    public string Descr { get => descr; set => descr = value; }
    [Column("condit")]
    [JsonProperty("Condit")]
    public string Condit { get => condit; set => condit = value;
}

    [Column("id_type")]
    [JsonProperty("IdType")]
    public int IdType { get => idType; set => idType = value; }
    [Column("id_dir")]
    [JsonProperty("IdDir")]
    public int IdDir { get => idDir; set => idDir = value; }
    [Column("outdated")]
    [DefaultValue(false)]
    [JsonProperty("Outdated")]

```

```

        public bool Outdated { get => outdated; set => outdated =
value; }
        [NotMapped]
        [JsonProperty("Current")]
        public string Current { get; set; }
    }

    [Table("Service_Age")]
    public class ServiceAge
    {
        private int idServ;
        private int idGroup;

        public ServiceAge()
        {
        }

        public ServiceAge(int idServ, int idGroup)
        {
            this.idServ = idServ;
            this.idGroup = idGroup;
        }

        [Key]
        [Column("id_serv")]
        [JsonProperty("IdServ")]
        public int IdServ { get => idServ; set => idServ = value; }
        [Key]
        [Column("id_group")]
        [JsonProperty("IdGroup")]
        public int IdGroup { get => idGroup; set => idGroup = value;
    }

        [NotMapped]
        [JsonProperty("Current")]
        public string Current { get; set; }
    }

    [Table("Status")]
    public class Status
    {
        private int idStat;
        private string name;

        public Status()
        {
        }

        public Status(int idStat, string name)
        {
            this.idStat = idStat;
            this.name = name;
        }
    }

```

```

    [Key]
    [Column("id_stat")]
    [JsonProperty("IdStat")]
    public int IdStat { get => idStat; set => idStat = value; }
    [Column("name")]
    [JsonProperty("Name")]
    public string Name { get => name; set => name = value; }
    [NotMapped]
    [JsonProperty("Current")]
    public string Current { get; set; }
}

[Table("Types")]
public class Type
{
    private int idType;
    private string name;
    private bool outdated;

    public Type()
    {
    }

    public Type(string name)
    {
        this.name = name;
    }

    public Type(string name, bool outdated)
    {
        this.name = name;
        this.outdated = outdated;
    }

    [Key]
    [Column("id_type")]
    [JsonProperty("IdType")]
    public int IdType { get => idType; set => idType = value; }
    [Column("name")]
    [JsonProperty("Name")]
    public string Name { get => name; set => name = value; }
    [Column("outdated")]
    [DefaultValue(false)]
    [JsonProperty("Outdated")]
    public bool Outdated { get => outdated; set => outdated =
value; }
    [NotMapped]
    [JsonProperty("Current")]
    public string Current { get; set; }
}

[Table("Users")]
public class User

```



```

{
    private string idEmp;
    private string login;
    private string password;
    private bool root;

    public User()
    {
    }

    public User(string login, string password)
    {
        this.login = login;
        this.password = password;
    }

    public User(string login, string password, bool root)
    {
        this.login = login;
        this.password = password;
        this.root = root;
    }

    [Key]
    [Column("id_emp")]
    [JsonProperty("IdEmp")]
    public string IdEmp { get => idEmp; set => idEmp = value; }
    [Column("login")]
    [JsonProperty("Login")]
    public string Login { get => login; set => login = value; }
    [Column("password")]
    [JsonProperty("Password")]
    public string Password { get => password; set => password =
value; }
    [Column("root")]
    [DefaultValue(false)]
    [JsonProperty("Root")]
    public bool Root { get => root; set => root = value; }
}

```

### **В.3 Методы контроллера для вывода данных**

```

[Route("[action]")]
[ApiController]
public partial class KidStrController : Controller
{
    KidstrContext db;
    public KidStrController(KidstrContext context)
    {
        db = context;
    }
    [HttpGet]

```

```

        public async Task<ActionResult<IEnumerable<AddServ>>>
AddServs() => await db.AddServs.ToListAsync();
        [HttpGet]
        public async Task<ActionResult<IEnumerable<AddServOrdr>>>
AddServOrdrs() => await db.AddServOrdrs.ToListAsync();
        [HttpGet]
        public async Task<ActionResult<IEnumerable<Age>>> Ages() =>
await db.Ages.ToListAsync();
        [HttpGet]
        public async Task<ActionResult<IEnumerable<Day>>> Days() =>
await db.Days.ToListAsync();
        [HttpGet]
        public async Task<ActionResult<IEnumerable<Direction>>>
Directions() => await db.Directions.ToListAsync();
        [HttpGet]
        public async Task<ActionResult<IEnumerable<Employee>>>
Employees() => await db.Employees.ToListAsync();
        [HttpGet]
        public async Task<ActionResult<IEnumerable<Guide>>> Guides()
=> await db.Guides.ToListAsync();
        [HttpGet]
        public async Task<ActionResult<IEnumerable<Ordr>>> Ordrs() =>
await db.Ordrs.ToListAsync();
        [HttpGet]
        public async Task<ActionResult<IEnumerable<Price>>> Prices()
=> await db.Prices.ToListAsync();
        [HttpGet]
        public async Task<ActionResult<IEnumerable<Service>>>
Services() => await db.Services.ToListAsync();
        [HttpGet]
        public async Task<ActionResult<IEnumerable<ServiceAge>>>
ServiceAges() => await db.ServiceAges.ToListAsync();
        [HttpGet]
        public async Task<ActionResult<IEnumerable<Status>>>
Statuses() => await db.Statuses.ToListAsync();
        [HttpGet]
        public async Task<ActionResult<IEnumerable<Models.Type>>>
Types() => await db.Types.ToListAsync();
    }

```

## **В.4 Методы контроллера для авторизации и добавления данных**

```

public partial class KidStrController : Controller
{
    //Authentication
    [HttpPost]
    public async Task<ActionResult<int>> Authent([FromBody] User
user)
    {
        if (user == null)
        {
            return -2;
        }
    }
}

```

```

        }
        User realuser = await db.Users.FirstOrDefaultAsync(x =>
(x.Login == user.Login) && (x.Password == user.Password));
        if (realuser == null)
            return -1;
        Funcs.Logs.Login(realuser.Login);
        if (realuser.Root)
            return 1;
        else
            return 0;
    }

    //Add data
    //Add Servs
    [HttpPost]
    public async Task<ActionResult<int>> AddAServ([FromBody]
AddServ data)
    {
        if (data == null)
        {
            return -2;
        }
        Funcs.Logs.AddLog(data.Current, "add", "AddServ");
        db.AddServs.Add(data);
        await db.SaveChangesAsync();
        return data.IdAdd;
    }

    //Add ServOrdrs
    [HttpPost]
    public async Task<ActionResult<int>> AddAServOrdr([FromBody]
AddServOrdr data)
    {
        if (data == null)
        {
            return -2;
        }
        db.AddServOrdrs.Add(data);
        await db.SaveChangesAsync();
        return 0;
    }

    //Ages
    [HttpPost]
    public async Task<ActionResult<int>> AddAge([FromBody] Age
data)
    {
        if (data == null)
        {
            return -2;
        }
        db.Ages.Add(data);
        await db.SaveChangesAsync();
    }

```

```

        return data.IdGroup;
    }

    //Days
    [HttpPost]
    public async Task<ActionResult<int>> AddDay([FromBody] Day
data)
    {
        if (data == null)
        {
            return -2;
        }
        db.Days.Add(data);
        await db.SaveChangesAsync();
        return 0;
    }

    //Directions
    [HttpPost]
    public async Task<ActionResult<int>> AddDirection([FromBody]
Direction data)
    {
        if (data == null)
        {
            return -2;
        }
        Funcs.Logs.AddLog(data.Current, "add", "Direction");
        db.Directions.Add(data);
        await db.SaveChangesAsync();
        return data.IdDir;
    }

    //Employees
    [HttpPost]
    public async Task<ActionResult<int>> AddEmployee([FromBody]
Employee data)
    {
        if (data == null)
        {
            return -2;
        }
        Funcs.Logs.AddLog(data.Current, "add", "Employee");
        db.Employees.Add(data);
        await db.SaveChangesAsync();
        return 0;
    }

    //Guides
    [HttpPost]
    public async Task<ActionResult<int>> AddGuide([FromBody]
Guide data)
    {
        if (data == null)

```

```

        {
            return -2;
        }
        db.Guides.Add(data);
        await db.SaveChangesAsync();
        return 0;
    }

    //Ordrs
    [HttpPost]
    public async Task<ActionResult<int>> AddOrdr([FromBody] Ordr
data)
    {
        if (data == null)
        {
            return -2;
        }
        Funcs.Logs.AddLog(data.Current, "add", "Ordr");
        db.Ordrs.Add(data);
        await db.SaveChangesAsync();
        return data.IdOrdr;
    }

    //Prices
    [HttpPost]
    public async Task<ActionResult<int>> AddPrice([FromBody]
Price data)
    {
        if (data == null)
        {
            return -2;
        }
        db.Prices.Add(data);
        await db.SaveChangesAsync();
        return data.IdPrice;
    }

    //Services
    [HttpPost]
    public async Task<ActionResult<int>> AddService([FromBody]
Service data)
    {
        if (data == null)
        {
            return -2;
        }
        Funcs.Logs.AddLog(data.Current, "add", "Service");
        db.Services.Add(data);
        await db.SaveChangesAsync();
        return data.IdServ;
    }

    //ServiceAges

```

```

        [HttpPost]
        public async Task<ActionResult<int>>
AddServiceAge([FromBody] ServiceAge data)
        {
            if (data == null)
            {
                return -2;
            }
            db.ServiceAges.Add(data);
            await db.SaveChangesAsync();
            return 0;
        }

//Statuses
[HttpPost]
public async Task<ActionResult<int>> AddStatus([FromBody]
Status data)
{
    if (data == null)
    {
        return -2;
    }
    db.Statuses.Add(data);
    await db.SaveChangesAsync();
    return data.IdStat;
}

//Types
[HttpPost]
public async Task<ActionResult<int>> AddType([FromBody]
Models.Type data)
{
    if (data == null)
    {
        return -2;
    }
    Funcs.Logs.AddLog(data.Current, "add", "Type");
    db.Types.Add(data);
    await db.SaveChangesAsync();
    return data.IdType;
}
}

```

## **B.5 Методы контроллера для изменения данных**

```

public partial class KidStrController : Controller
{
    //AddServs
    [HttpPut]
    public async Task<ActionResult<int>> UpdateAServ([FromBody]
AddServ data)
    {

```

```

        if (data == null)
        {
            return -2;
        }
        if (!db.AddServs.Any(x => x.IdAdd == data.IdAdd))
        {
            return -1;
        }
        Funcs.Logs.AddLog(data.Current, "update", "AddServ");
        db.Update(data);
        await db.SaveChangesAsync();
        return 0;
    }

    //AddServOrdrs
    [HttpPut]
    public async Task<ActionResult<int>>
UpdateAServOrdr([FromBody] AddServOrdr data)
    {
        if (data == null)
        {
            return -2;
        }
        if (!db.AddServOrdrs.Any(x => x.IdAdd == data.IdAdd &&
x.IdOrdr == data.IdOrdr))
        {
            return -1;
        }
        db.Update(data);
        await db.SaveChangesAsync();
        return 0;
    }

    //Ages
    [HttpPut]
    public async Task<ActionResult<int>> UpdateAge([FromBody] Age
data)
    {
        if (data == null)
        {
            return -2;
        }
        if (!db.Ages.Any(x => x.IdGroup == data.IdGroup))
        {
            return -1;
        }
        db.Update(data);
        await db.SaveChangesAsync();
        return 0;
    }

    //Directions
    [HttpPut]

```

```

        public async Task<ActionResult<int>>
UpdateDirection([FromBody] Direction data)
{
    if (data == null)
    {
        return -2;
    }
    if (!db.Directions.Any(x => x.IdDir == data.IdDir))
    {
        return -1;
    }
    Funcs.Logs.AddLog(data.Current, "update", "Direction");
    db.Update(data);
    await db.SaveChangesAsync();
    return 0;
}

//Employees
[HttpPut]
public async Task<ActionResult<int>>
UpdateEmployee([FromBody] Employee data)
{
    if (data == null)
    {
        return -2;
    }
    if (!db.Employees.Any(x => x.IdEmp == data.IdEmp))
    {
        return -1;
    }
    Funcs.Logs.AddLog(data.Current, "update", "Employee");
    db.Update(data);
    await db.SaveChangesAsync();
    return 0;
}

//Ordrs
[HttpPut]
public async Task<ActionResult<int>> UpdateOrdr([FromBody]
Ordr data)
{
    if (data == null)
    {
        return -2;
    }
    if (!db.Ordrs.Any(x => x.IdOrdr == data.IdOrdr))
    {
        return -1;
    }
    Funcs.Logs.AddLog(data.Current, "update", "Ordr");
    db.Update(data);
    await db.SaveChangesAsync();
    return 0;
}

```



```

    }

    //Prices
    [HttpPut]
    public async Task<ActionResult<int>> UpdatePrice([FromBody]
Price data)
    {
        if (data == null)
        {
            return -2;
        }
        if (!db.Prices.Any(x => x.IdPrice == data.IdPrice))
        {
            return -1;
        }
        db.Update(data);
        await db.SaveChangesAsync();
        return 0;
    }

    //Services
    [HttpPut]
    public async Task<ActionResult<int>>
UpdateService([FromBody] Service data)
    {
        if (data == null)
        {
            return -2;
        }
        if (!db.Services.Any(x => x.IdServ == data.IdServ))
        {
            return -1;
        }
        Funcs.Logs.AddLog(data.Current, "update", "Service");
        db.Update(data);
        await db.SaveChangesAsync();
        return 0;
    }

    //Statuses
    [HttpPut]
    public async Task<ActionResult<int>> UpdateStatus([FromBody]
Status data)
    {
        if (data == null)
        {
            return -2;
        }
        if (!db.Statuses.Any(x => x.IdStat == data.IdStat))
        {
            return -1;
        }
        db.Update(data);
    }

```

```

        await db.SaveChangesAsync();
        return 0;
    }

    //Types
    [HttpPut]
    public async Task<ActionResult<int>> UpdateType([FromBody]
Models.Type data)
    {
        if (data == null)
        {
            return -2;
        }
        if (!db.Types.Any(x => x.IdType == data.IdType))
        {
            return -1;
        }
        Funcs.Logs.AddLog(data.Current, "update", "Type");
        db.Update(data);
        await db.SaveChangesAsync();
        return 0;
    }
}

```

## **В.6 Методы контроллера для удаления данных**

```

public partial class KidStrController : Controller
{
    //AddServs
    [HttpDelete("{id}/{user}")]
    public async Task<ActionResult<int>> DeleteAServ(int id,
string user)
    {
        AddServ data = db.AddServs.First(x => x.IdAdd == id);
        if (data == null)
        {
            return -1;
        }
        Funcs.Logs.AddLog(user, "delete", "AddServ");
        if (db.AddServOrdrs.Any(x => x.IdAdd == id))
            data.Outdated = true;
        else
            db.AddServs.Remove(data);
        await db.SaveChangesAsync();
        return 0;
    }

    //AddServOrdrs
    [HttpDelete("{idS}/{idO}")]
    public async Task<ActionResult<int>> DeleteAServOrdr(int idS,
int idO)
    {

```

```

        AddServOrdr data = db.AddServOrdrs.First(x => (x.IdAdd
== idS && x.IdOrdr == idO));
        if (data == null)
        {
            return -1;
        }
        db.AddServOrdrs.Remove(data);
        await db.SaveChangesAsync();
        return 0;
    }

    //Ages
    [HttpDelete("{id}")]
    public async Task<ActionResult<int>> DeleteAge(int id)
    {
        Age data = db.Ages.First(x => x.IdGroup == id);
        if (data == null)
        {
            return -1;
        }
        db.Ages.Remove(data);
        await db.SaveChangesAsync();
        return 0;
    }

    //Days
    [HttpDelete("{idS}/{idN}")]
    public async Task<ActionResult<int>> DeleteDay(int idS, int
idN)
    {
        Day data = db.Days.First(x => (x.IdServ == idS &&
x.DayNum == idN));
        if (data == null)
        {
            return -1;
        }
        db.Days.Remove(data);
        await db.SaveChangesAsync();
        return 0;
    }

    //Directions
    [HttpDelete("{id}/{user}")]
    public async Task<ActionResult<int>> DeleteDirection(int id,
string user)
    {
        Direction data = db.Directions.First(x => x.IdDir ==
id);
        if (data == null)
        {
            return -1;
        }
        Funcs.Logs.AddLog(user, "delete", "Direction");
    }

```

```

        if (db.Services.Any(x => x.IdDir == id))
            data.Outdated = true;
        else
            db.Directions.Remove(data);
        await db.SaveChangesAsync();
        return 0;
    }

    //Employees
    [HttpDelete("{id}/{user}")]
    public async Task<ActionResult<int>> DeleteEmployee(string
id, string user)
    {
        Employee data = db.Employees.First(x => x.IdEmp == id);
        if (data == null)
        {
            return -1;
        }
        Funcs.Logs.AddLog(user, "delete", "Employee");
        db.Employees.Remove(data);
        await db.SaveChangesAsync();
        return 0;
    }

    //Guides
    [HttpDelete("{idE}/{idS}")]
    public async Task<ActionResult<int>> DeleteGuide(string idE,
int idS)
    {
        Guide data = db.Guides.First(x => (x.IdEmp == idE &&
x.IdServ == idS));
        if (data == null)
        {
            return -1;
        }
        db.Guides.Remove(data);
        await db.SaveChangesAsync();
        return 0;
    }

    //Ordrs
    [HttpDelete("{id}/{user}")]
    public async Task<ActionResult<int>> DeleteOrdr(int id,
string user)
    {
        Ordr data = db.Ordrs.First(x => x.IdOrdr == id);
        if (data == null)
        {
            return -1;
        }
        Funcs.Logs.AddLog(user, "delete", "Ordr");
        db.Ordrs.Remove(data);
        await db.SaveChangesAsync();
    }

```

```

        return 0;
    }

    //Prices
    [HttpDelete("{id}")]
    public async Task<ActionResult<int>> DeletePrice(int id)
    {
        Price data = db.Prices.First(x => x.IdPrice == id);
        if (data == null)
        {
            return -1;
        }
        if (db.Orders.Any(x => x.IdPrice == id))
            data.Outdated = true;
        else
            db.Prices.Remove(data);
        await db.SaveChangesAsync();
        return 0;
    }

    //Services
    [HttpDelete("{id}/{user}")]
    public async Task<ActionResult<int>> DeleteService(int id,
string user)
    {
        Service data = db.Services.First(x => x.IdServ == id);
        if (data == null)
        {
            return -1;
        }
        Funcs.Logs.AddLog(user, "delete", "Service");
        if ((from s in db.Services
            join p in db.Prices on s.IdServ equals p.IdServ
            join o in db.Orders on p.IdPrice equals o.IdPrice
            where s.IdServ == id
            select o).Count()>0)
            data.Outdated = true;
        else
            db.Services.Remove(data);
        await db.SaveChangesAsync();
        return 0;
    }

    //ServiceAges
    [HttpDelete("{idS}/{idG}")]
    public async Task<ActionResult<int>> DeleteServiceAge(int
idS, int idG)
    {
        ServiceAge data = db.ServiceAges.First(x => (x.IdServ ==
idS && x.IdGroup == idG));
        if (data == null)
        {
            return -1;
        }
    }

```

```

        }
        db.ServiceAges.Remove(data);
        await db.SaveChangesAsync();
        return 0;
    }

    //Statuses
    [HttpDelete("{id}")]
    public async Task<ActionResult<int>> DeleteStatus(int id)
    {
        Status data = db.Statuses.First(x => x.IdStat == id);
        if (data == null)
        {
            return -1;
        }
        db.Statuses.Remove(data);
        await db.SaveChangesAsync();
        return 0;
    }

    //Types
    [HttpDelete("{id}/{user}")]
    public async Task<ActionResult<int>> DeleteType(int id,
string user)
    {
        Models.Type data = db.Types.First(x => x.IdType == id);
        if (data == null)
        {
            return -1;
        }
        Funcs.Logs.AddLog(user, "delete", "Type");
        if (db.Services.Any(x => x.IdType == id))
            data.Outdated = true;
        else
            db.Types.Remove(data);
        await db.SaveChangesAsync();
        return 0;
    }
}

```

## **В.7 Класс функций журналирования**

```

public static class Logs
{
    public static void AddLog(string current, string action,
string table)
    {
        Directory.CreateDirectory("Logs");
        using (StreamWriter writer =
File.AppendText("Logs/logfile.txt"))
        {
            writer.WriteLine($"{DateTime.UtcNow.AddHours(3)} :

```

```

User {current} {action} 1 row to {table}");
    }
}
public static void Login(string current)
{
    Directory.CreateDirectory("Logs");
    using (StreamWriter writer =
File.AppendText("Logs/logfile.txt"))
    {
        writer.WriteLine($"{DateTime.UtcNow.AddHours(3)} :
User {current} log in");
    }
}
}

```

## ПРИЛОЖЕНИЕ Г

### Код основных элементов пользовательского интерфейса

#### Г.1 Код модели со стороны пользовательского интерфейса

```
public class AddServ
{
    private int idAdd;
    private string name;
    private double cost;
    private bool outdated;

    public AddServ()
    {
    }

    public AddServ(string name, double cost, bool outdated)
    {
        this.name = name;
        this.cost = cost;
        this.outdated = outdated;
    }

    public AddServ(string name, double cost)
    {
        this.name = name;
        this.cost = cost;
    }

    [JsonProperty("IdAdd")]
    public int IdAdd { get => idAdd; set => idAdd = value; }
    [JsonProperty("Name")]
    public string Name { get => name; set => name = value; }
    [DefaultValue(0)]
    [JsonProperty("Cost")]
    public double Cost { get => cost; set => cost = value; }
    [DefaultValue(false)]
    [JsonProperty("Outdated")]
    public bool Outdated { get => outdated; set => outdated =
value; }
    [JsonIgnore]
    public int Count { get; set; }
    [Browsable(false)]
    [JsonProperty("Current")]
    public string Current { get; set; }
}

public class AddServOrder
{

```



```

private int idAdd;
private int idOrdr;
private int addCount;

public AddServOrdr()
{
}

public AddServOrdr(int idAdd, int idOrdr, int addCount)
{
    this.idAdd = idAdd;
    this.idOrdr = idOrdr;
    this.addCount = addCount;
}

public AddServOrdr(int idAdd, int idOrdr)
{
    this.idAdd = idAdd;
    this.idOrdr = idOrdr;
}

[JsonProperty("IdAdd")]
public int IdAdd { get => idAdd; set => idAdd = value; }
[JsonProperty("IdOrdr")]
public int IdOrdr { get => idOrdr; set => idOrdr = value; }
[DefaultValue(1)]
[JsonProperty("AddCount")]
public int AddCount { get => addCount; set => addCount =
value; }
[Browsable(false)]
[JsonProperty("Current")]
public string Current { get; set; }
}

public class Age
{
    private int idGroup;
    private string name;

    public Age()
    {
    }

    public Age(string name)
    {
        this.name = name;
    }

    [JsonProperty("IdGroup")]
    public int IdGroup { get => idGroup; set => idGroup = value;
}

    [JsonProperty("Name")]
    public string Name { get => name; set => name = value; }
    [Browsable(false)]

```

```

        [JsonProperty("Current")]
        public string Current { get; set; }
    }

    public class Day
    {
        private int idServ;
        private int dayNum;

        public Day()
        {
        }

        public Day(int idServ, int dayNum)
        {
            this.idServ = idServ;
            this.dayNum = dayNum;
        }

        [JsonProperty("IdServ")]
        public int IdServ { get => idServ; set => idServ = value; }
        [JsonProperty("DayNum")]
        public int DayNum { get => dayNum; set => dayNum = value; }
        [Browsable(false)]
        [JsonProperty("Current")]
        public string Current { get; set; }
    }

    public class Direction
    {
        private int idDir;
        private string name;
        private bool outdated;

        public Direction()
        {
        }

        public Direction(string name, bool outdated)
        {
            this.name = name;
            this.outdated = outdated;
        }

        public Direction(string name)
        {
            this.name = name;
        }

        public Direction(int idDir, string name)
        {
            this.idDir = idDir;
            this.name = name;
        }
    }

```

```

    }

    [JsonProperty("IdDir")]
    public int IdDir { get => idDir; set => idDir = value; }
    [JsonProperty("Name")]
    public string Name { get => name; set => name = value; }
    [DefaultValue(false)]
    [JsonProperty("Outdated")]
    public bool Outdated { get => outdated; set => outdated =
value; }
    [Browsable(false)]
    [JsonProperty("Current")]
    public string Current { get; set; }
}

public class Employee
{
    private string idEmp;
    private string name;
    private string phone;

    public Employee()
    {
    }

    public Employee(string idEmp, string name, string phone)
    {
        this.idEmp = idEmp;
        this.name = name;
        this.phone = phone;
    }

    [JsonProperty("IdEmp")]
    public string IdEmp { get => idEmp; set => idEmp = value; }
    [JsonProperty("Name")]
    public string Name { get => name; set => name = value; }
    [JsonProperty("Phone")]
    public string Phone { get => phone; set => phone = value; }
    [JsonIgnore]
    public List<Service> Guide { get; set; }
    [Browsable(false)]
    [JsonProperty("Current")]
    public string Current { get; set; }
    [JsonIgnore]
    public List<Order> Orders { get; set; }
    public override string ToString() { return $"{IdEmp} | {Name}
| {Phone}"; }
}

public class Guide
{
    private string idEmp;
    private int idServ;

```

```

public Guide()
{
}

public Guide(string idEmp, int idServ)
{
    this.idEmp = idEmp;
    this.idServ = idServ;
}

[JsonProperty("IdEmp")]
public string IdEmp { get => idEmp; set => idEmp = value; }
[JsonProperty("IdServ")]
public int IdServ { get => idServ; set => idServ = value; }
[Browsable(false)]
[JsonProperty("Current")]
public string Current { get; set; }
}

public class Ordr
{
    private int idOrdr;
    private string school;
    private int idGroup;
    private int partyCount;
    private int accomCount;
    private string contact;
    private DateTime dateTime;
    private string meetPoint;
    private double? cost;
    private double? prepay;
    private int idPrice;
    private int idStat;
    private string idEmp;

    public Ordr()
    {
    }

    [JsonProperty("IdOrdr")]
    public int IdOrdr { get => idOrdr; set => idOrdr = value; }
    [JsonProperty("School")]
    public string School { get => school; set => school = value;
}

    [Browsable(false)]
    [JsonProperty("IdGroup")]
    public int IdGroup { get => idGroup; set => idGroup = value;
}

    [JsonIgnore]
    public string AgeName { get => Age.Name; }
    [JsonProperty("PartyCount")]
    public int PartyCount { get => partyCount; set => partyCount

```

```

= value; }
    [DefaultValue(0)]
    [JsonProperty("AccomCount")]
    public int AccomCount { get => accomCount; set => accomCount
= value; }
    [JsonProperty("Contact")]
    public string Contact { get => contact; set => contact =
value; }
    [JsonProperty("DateTime")]
    public DateTime DateTime { get => dateTime; set => dateTime
= value; }
    [JsonProperty("MeetPoint")]
    public string MeetPoint { get => meetPoint; set => meetPoint
= value; }
    [JsonProperty("Cost")]
    public double? Cost { get => cost; set => cost = value; }
    [JsonProperty("Prepay")]
    public double? Prepay { get => prepay; set => prepay = value;
}

    [JsonIgnore]
    public string ServiceName { get => Service.Name; }
    [Browsable(false)]
    [JsonProperty("IdPrice")]
    public int IdPrice { get => idPrice; set => idPrice = value;
}

    [Browsable(false)]
    [JsonProperty("IdStat")]
    public int IdStat { get => idStat; set => idStat = value; }
    [Browsable(false)]
    [JsonProperty("IdEmp")]
    public string IdEmp { get => idEmp; set => idEmp = value; }
    [JsonIgnore]
    public string Status { get; set; }
    [Browsable(false)]
    [JsonIgnore]
    public Service Service { get; set; }
    [Browsable(false)]
    [JsonIgnore]
    public Price Price { get; set; }
    [Browsable(false)]
    [JsonIgnore]
    public Age Age { get; set; }
    [Browsable(false)]
    [JsonIgnore]
    public Employee Employee { get; set; }
    [JsonIgnore]
    public List<AddServ> AddServs { get; set; }
    [Browsable(false)]
    [JsonProperty("Current")]
    public string Current { get; set; }
}

public class Price

```

```

{
    private int idPrice;
    private int idServ;
    private int memberMin;
    private int? memberMax;
    private int accomCount;
    private string conduit;
    private double cost;
    private bool outdated;

    public Price()
    {
    }

    public Price(int idServ, int memberMin, int? memberMax, int
accomCount, string conduit, double cost, bool outdated)
    {
        this.idServ = idServ;
        this.memberMin = memberMin;
        this.memberMax = memberMax;
        this.accomCount = accomCount;
        this.conduit = conduit;
        this.cost = cost;
        this.outdated = outdated;
    }

    [JsonProperty("IdPrice")]
    public int IdPrice { get => idPrice; set => idPrice = value;
}

    [Browsable(false)]
    [JsonProperty("IdServ")]
    public int IdServ { get => idServ; set => idServ = value; }
    [DefaultValue(0)]
    [JsonProperty("MemberMin")]
    public int MemberMin { get => memberMin; set => memberMin =
value; }
    [JsonProperty("MemberMax")]
    public int? MemberMax { get => memberMax; set => memberMax =
value; }
    [DefaultValue(0)]
    [JsonProperty("AccomCount")]
    public int AccomCount { get => accomCount; set => accomCount
= value; }
    [JsonProperty("Conduit")]
    public string Conduit { get => conduit; set => conduit = value;
}

    [DefaultValue(0)]
    [JsonProperty("Cost")]
    public double Cost { get => cost; set => cost = value; }
    [DefaultValue(false)]
    [JsonProperty("Outdated")]
    public bool Outdated { get => outdated; set => outdated =
value; }
}

```

```

[Browsable(false)]
[JsonProperty("Current")]
public string Current { get; set; }
public override string ToString()
{
    string max = MemberMax.HasValue ? MemberMax.ToString()
: "...";
    return $"{IdPrice} | {MemberMin}-{max} чел.,
{AccomCount} comp. | {Condit} | {Cost} p.";
}

}

public class Service
{
    private int idServ;
    private string name;
    private string duration;
    private string descr;
    private string condit;
    private int idType;
    private int idDir;
    private bool outdated;

    public Service()
    {
    }

    public Service(string name, string duration, string descr,
string condit, int idType, int idDir)
    {
        this.name = name;
        this.duration = duration;
        this.descr = descr;
        this.condit = condit;
        this.idType = idType;
        this.idDir = idDir;
    }

    public Service(string name, string duration, string descr,
string condit, int idType, int idDir, bool outdated)
    {
        this.name = name;
        this.duration = duration;
        this.descr = descr;
        this.condit = condit;
        this.idType = idType;
        this.idDir = idDir;
        this.outdated = outdated;
    }

    [JsonProperty("IdServ")]
    public int IdServ { get => idServ; set => idServ = value; }
    [JsonProperty("Name")]

```

```

        public string Name { get => name; set => name = value; }
        [JsonProperty("Duration")]
        public string Duration { get => duration; set => duration =
value; }
        [JsonProperty("Descr")]
        public string Descr { get => descr; set => descr = value; }
        [JsonProperty("Condit")]
        public string Condit { get => condit; set => condit = value;
}

        [JsonProperty("IdType")]
        public int IdType { get => idType; set => idType = value; }
        [JsonProperty("IdDir")]
        public int IdDir { get => idDir; set => idDir = value; }
        [DefaultValue(false)]
        [JsonProperty("Outdated")]
        public bool Outdated { get => outdated; set => outdated =
value; }
        [JsonIgnore]
        public string Type { get; set; }
        [JsonIgnore]
        public string Direction { get; set; }
        [JsonIgnore]
        public List<Price> Prices { get; set; }
        [JsonIgnore]
        public List<Day> Days { get; set; }
        [JsonIgnore]
        public List<Age> Ages { get; set; }
        [JsonIgnore]
        public bool Checked { get; set; }
        [Browsable(false)]
        [JsonProperty("Current")]
        public string Current { get; set; }
        public override string ToString() { return $"{IdServ} | {Type}
| {Duration} | {Descr} | {Condit}"; }

}

public class ServiceAge
{
    private int idServ;
    private int idGroup;

    public ServiceAge()
    {
    }

    public ServiceAge(int idServ, int idGroup)
    {
        this.idServ = idServ;
        this.idGroup = idGroup;
    }

    [JsonProperty("IdServ")]

```



```

    public int IdServ { get => idServ; set => idServ = value; }
    [JsonProperty("IdGroup")]
    public int IdGroup { get => idGroup; set => idGroup = value;
}
    [Browsable(false)]
    [JsonProperty("Current")]
    public string Current { get; set; }
}

public class Status
{
    private int idStat;
    private string name;

    public Status()
    {
    }

    public Status(int idStat, string name)
    {
        this.idStat = idStat;
        this.name = name;
    }

    [JsonProperty("IdStat")]
    public int IdStat { get => idStat; set => idStat = value; }
    [JsonProperty("Name")]
    public string Name { get => name; set => name = value; }
    [Browsable(false)]
    [JsonProperty("Current")]
    public string Current { get; set; }
}

public class Type
{
    private int idType;
    private string name;
    private bool outdated;

    public Type()
    {
    }

    public Type(string name)
    {
        this.name = name;
    }

    public Type(string name, bool outdated)
    {
        this.name = name;
        this.outdated = outdated;
    }
}

```

```

public Type(int idType, string name)
{
    this.idType = idType;
    this.name = name;
}

[JsonProperty("IdType")]
public int IdType { get => idType; set => idType = value; }
[JsonProperty("Name")]
public string Name { get => name; set => name = value; }
[DefaultValue(false)]
[JsonProperty("Outdated")]
public bool Outdated { get => outdated; set => outdated =
value; }
[Browsable(false)]
[JsonProperty("Current")]
public string Current { get; set; }
}

public class User
{
    private string idEmp;
    private string login;
    private string password;
    private bool root;

    public User()
    {
    }

    public User(string login, string password)
    {
        this.login = login;
        this.password = password;
    }

    public User(string login, string password, bool root)
    {
        this.login = login;
        this.password = password;
        this.root = root;
    }

    [JsonProperty("IdEmp")]
    public string IdEmp { get => idEmp; set => idEmp = value; }
    [JsonProperty("Login")]
    public string Login { get => login; set => login = value; }
    [JsonProperty("Password")]
    public string Password { get => password; set => password =
value; }
    [DefaultValue(false)]
    [JsonProperty("Root")]

```

```

    public bool Root { get => root; set => root = value; }
}

```

## Г.2 Класс функций взаимодействия с Web API

```

public static class JsonFunc
{
    private static string site =
@"http://kidstrWebApi.somee.com/";

    public static string Request(string name, string method,
string data)
    {
        HttpWebRequest request = WebRequest.Create(site + name)
as HttpWebRequest;
        request.Method = method.ToUpper(); //POST, PUT
        request.ContentType = "application/json; charset=utf-
8";
        using (StreamWriter writer = new
StreamWriter(request.GetRequestStream()))
        {
            writer.Write(data);
        }
        WebResponse response = request.GetResponse();
        Stream stream = response.GetResponseStream();
        StreamReader reader = new StreamReader(stream);
        return reader.ReadToEnd();
    }

    public static string ReturnList(string name)
    {
        HttpWebRequest request = WebRequest.Create(site + name)
as HttpWebRequest;
        request.Method = "GET";
        WebResponse response = request.GetResponse();
        Stream stream = response.GetResponseStream();
        StreamReader reader = new StreamReader(stream);
        return reader.ReadToEnd();
    }

    public static string Delete(string name, string data)
    {
        HttpWebRequest request = WebRequest.Create(site + name
+ "/" + data) as HttpWebRequest;
        request.Method = "DELETE";
        WebResponse response = request.GetResponse();
        Stream stream = response.GetResponseStream();
        StreamReader reader = new StreamReader(stream);
        return reader.ReadToEnd();
    }
}

```

### Г.3 Класс текущего пользователя

```
public static class CurrentUser
{
    public static User current;
    public static string HashPassword(string toEncrypt)
    {
        using (SHA256 sha256Hash = SHA256.Create())
        {
            byte[] bytes =
sha256Hash.ComputeHash(Encoding.UTF8.GetBytes(toEncrypt));
            StringBuilder builder = new StringBuilder();
            for (int i = 0; i < bytes.Length; i++)
            {
                builder.Append(bytes[i].ToString("x2"));
            }
            return builder.ToString();
        }
    }
}
```

### Г.4 Класс функций преобразования таблиц

```
public static class TableFunc
{
    public static DataTable ToDataTable<T>(this IEnumerable<T>
collection)
    {
        DataTable dt = new DataTable();
        Type t = typeof(T);
        PropertyInfo[] pia = t.GetProperties();
        object temp;
        DataRow dr;

        for (int i = 0; i < pia.Length; i++)
        {
            if (! (pia[i].PropertyType.IsGenericType &&
pia[i].PropertyType.IsGenericTypeDefinition))
            {
                dt.Columns.Add(pia[i].Name,
Nullable.GetUnderlyingType(pia[i].PropertyType) ??
pia[i].PropertyType);
                dt.Columns[i].AllowDBNull = true;
            }
        }
        foreach (T item in collection)
        {
            dr = dt.NewRow();
            dr.BeginEdit();

            for (int i = 0; i < pia.Length; i++)
```

```

        {
            if (!(pia[i].PropertyType.IsGenericType &&
pia[i].PropertyType.IsGenericTypeDefinition))
            {
                temp = pia[i].GetValue(item, null);
                if (temp == null || (temp.GetType().Name
== "Char" && ((char)temp).Equals('\0')))
                {
                    dr[pia[i].Name] = DBNull.Value;
                }
                else
                {
                    dr[pia[i].Name] = temp;
                }
            }
        }

        dr.EndEdit();
        dt.Rows.Add(dr);
    }
    return dt;
}

public static DataGridView AddHeader(DataGridView table,
List<string> names)
{
    for (int i = 0; i < table.ColumnCount; i++)
    {
        try
        {
            table.Columns[i].HeaderText = names[i];
        }
        catch
        {
            table.Columns[i].HeaderText = "";
        }
    }
    return table;
}
}

```

## Г.5 Окно авторизации

```

public partial class Authorization : Form
{
    public Authorization()
    {
        InitializeComponent();
    }
    private void Authorization_Load(object sender, EventArgs e)
    {

```

```

        CurrentUser.current = new User();
    }

    private void go_Click(object sender, EventArgs e)
    {
        CurrentUser.current.Login = "NaN";
        CurrentUser.current.Login = login.Text;
        CurrentUser.current.Password = password.Text;
        CurrentUser.HashPassword(password.Text);
        int result = int.Parse(JsonFunc.Request("Authent",
"POST", JsonConvert.SerializeObject(CurrentUser.current)));
        if (result == -1)
            MessageBox.Show("Неверный логин/пароль");
        else
            if (result == 0 && admin.Checked)
                MessageBox.Show("Нет прав доступа к панели администратора");
            else
                if (admin.Checked)
                {
                    MainAdmin main = new MainAdmin(this);
                    Hide();
                    main.Show();
                    login.Text = "";
                    password.Text = "";
                    admin.Checked = false;
                }
            else
            {
                MainManager main = new MainManager(this);
                Hide();
                main.Show();
                login.Text = "";
                password.Text = "";
                admin.Checked = false;
            }
    }
}

```

## Г.6 Главное окно менеджера

```

public partial class MainManager : Form
{
    private Form parent;
    private Form child;
    private bool backcommand;

    public MainManager(Form p)
    {
        parent = p;
        InitializeComponent();
    }
}

```

```

        private void MainManager_FormClosed(object sender,
FormClosedEventArgs e)
        {
            if (!backcommand)
            {
                Application.Exit();
            }
        }

        private void exit_Click(object sender, EventArgs e)
        {
            backcommand = true;
            Close();
            parent.Show();
        }

        private void menuAdminServ_Click(object sender, EventArgs e)
        {
            child = new ListService(this);
            Hide();
            child.Show();
        }

        private void menuAdminAddServ_Click(object sender, EventArgs
e)
        {
            child = new ListAServ(this);
            Hide();
            child.Show();
        }

        private void menuAdminOrders_Click(object sender, EventArgs
e)
        {
            child = new ListOrder(this);
            Hide();
            child.Show();
        }
    }
}

```

## Г.7 Главное окно администратора

```

public partial class MainAdmin : Form
{
    private Form parent;
    private Form child;
    private bool backcommand;

    public MainAdmin(Form p)
    {
        parent = p;
    }
}

```

```

        InitializeComponent();
    }
    private void MainAdmin_FormClosed(object sender,
FormClosedEventArgs e)
    {
        if (!backcommand)
        {
            Application.Exit();
        }
    }

    private void exit_Click(object sender, EventArgs e)
    {
        backcommand = true;
        Close();
        parent.Show();
    }

    private void menuAdminParamType_Click(object sender,
EventArgs e)
    {
        child = new ListType(this);
        Hide();
        child.Show();
    }

    private void menuAdminParamDirect_Click(object sender,
EventArgs e)
    {
        child = new ListDirection(this);
        Hide();
        child.Show();
    }

    private void menuAdminUsers_Click(object sender, EventArgs e)
    {
        child = new ListEmployee(this);
        Hide();
        child.Show();
    }

    private void menuAdminReports_Click(object sender, EventArgs
e)
    {
        child = new Reports(this);
        Hide();
        child.Show();
    }
}

```



## Г.8 Окно списка дополнительных услуг

```
public partial class ListAServ : Form
{
    private Form parent;
    private Form child;
    private bool backcommand;
    private string state;
    private List<AddServ> data;

    public ListAServ(Form p)
    {
        parent = p;
        InitializeComponent();
    }

    private void ListAServ_Load(object sender, EventArgs e)
    {
        Load_data();
        gridAddServ = TableFunc.AddHeader(gridAddServ, new
List<string> { "ID", "Название", "Стоимость", "Устарела" });
        gridAddServ.Columns[4].Visible = false;
    }

    private void ListAServ_FormClosed(object sender,
FormClosedEventArgs e)
    {
        if (!backcommand)
        {
            Application.Exit();
        }
    }

    private void refresh_Click(object sender, EventArgs e)
    {
        Load_data();
    }

    private void back_Click(object sender, EventArgs e)
    {
        backcommand = true;
        Close();
        parent.Show();
    }

    private void add_Click(object sender, EventArgs e)
    {
        child = new AddChangeAServ();
        child.ShowDialog();
        Load_data();
    }
}
```

```

private void change_Click(object sender, EventArgs e)
{
    child = new
AddChangeAServ((AddServ)gridAddServ.CurrentRow.DataBoundItem);
    child.ShowDialog();
    Load_data();
}

private void delete_Click(object sender, EventArgs e)
{
    DialogResult result = MessageBox.Show("Вы уверены, что
хотите удалить строку?", "Удалить", MessageBoxButtons.YesNo);
    if (result == DialogResult.Yes)
    {
        state = JsonFunc.Delete("DeleteAServ",
((AddServ)gridAddServ.CurrentRow.DataBoundItem).IdAdd.ToString()
+ "/" + CurrentUser.current.Login);
        if (state == "-1")
        {
            MessageBox.Show("Ошибка доступа к данным.
Пожалуйста, обратитесь к администратору.");
        }

        Load_data();
    }
}

private void Load_data()
{
    data =
JsonConvert.DeserializeObject<List<AddServ>>(JsonFunc.ReturnList
("AddServs"));
    gridAddServ.DataSource = new BindingSource(data, null);
    if (gridAddServ.Rows.Count == 0)
    {
        change.Enabled = false;
        delete.Enabled = false;
    }
    else
    {
        change.Enabled = true;
        delete.Enabled = true;
    }
}
}

```

## Г.9 Окно добавления, изменения и деталей заказа

```

public partial class AddChangeDetOrdr : Form
{
    private Ordr dataO;
    private bool change;
}

```

```

private string state;
private double? currentCost;
private Dictionary<int, int> oldAS = new Dictionary<int,
int>();
private List<AddServ> dataAS;
private List<Employee> dataE;
private List<Age> dataA;
private List<Service> dataS;
private List<Price> AvailableP = new List<Price>();
private List<Service> AvailableS = new List<Service>();
private List<Employee> AvailableE = new List<Employee>();

public AddChangeDetOrdr(List<AddServ> addServs,
List<Employee> employees, List<Age> ages, List<Service> services)
{
    dataO = new Ordr();
    dataAS = addServs;
    dataE = employees;
    dataA = ages;
    dataS = services;
    dataO.AddServs = new List<AddServ>();

    InitializeComponent();
    Text += "Добавить заказ";
}

public AddChangeDetOrdr(Ordr ordr, List<AddServ> addServs,
List<Employee> employees, List<Age> ages, List<Service> services)
{
    dataO = ordr;
    dataAS = addServs;
    dataE = employees;
    dataA = ages;
    dataS = services;

    change = true;

    InitializeComponent();

    Text += $"Изменить заказ № {dataO.IdOrdr}";
}

public AddChangeDetOrdr(Ordr ordr)
{
    dataO = ordr;
    dataA = new List<Age> { dataO.Age };
    dataAS = dataO.AddServs;
    dataE = new List<Employee> { dataO.Employee };
    dataS = new List<Service> { dataO.Service };

    InitializeComponent();

    Text += $"Детали заказа № {dataO.IdOrdr}";
    save.Visible = false;
    gridAServ.ReadOnly = true;
}

```

```

        school.ReadOnly = true;
        contact.ReadOnly = true;
        datetime.ReadOnly = true;
        meetpoint.ReadOnly = true;
        party.ReadOnly = true;
        accom.ReadOnly = true;
        party.Increment = 0;
        accom.Increment = 0;
    }

    private void AddChangeOrdr_Load(object sender, EventArgs e)
    {
        age.DataSource = dataA;
        age.DisplayMember = "Name";
        age.ValueMember = "IdGroup";
        if (dataO.IdOrdr > 0)
        {
            school.Text = dataO.School;
            contact.Text = dataO.Contact;
            datetime.Text = dataO.DateTime.ToString("dd.MM.yyyy HH:mm");
            meetpoint.Text = dataO.MeetPoint;
            party.Value = dataO.PartyCount;
            accom.Value = dataO.AccomCount;
            currentCost = dataO.Cost;
            cost.Text = currentCost.ToString();
            prepay.Text = dataO.Prepay.ToString();
            status.Text = dataO.Status;
            find_service();
            find_price();
            find_emp();
            age.SelectedValue = dataO.IdGroup;
            serv.DataSource = new BindingSource(new
List<Service> { dataO.Service }.Union(AvailableS), null);
            price.DataSource = new BindingSource(new
List<Price> { dataO.Price }.Union(AvailableP), null);
            guide.DataSource = new BindingSource(new
List<Employee> { dataO.Employee }.Union(AvailableE), null);
            serv.SelectedItem = dataO.Service;
            price.SelectedItem = dataO.Price;
            guide.SelectedItem = dataO.Employee;
        }

        foreach (AddServ a in dataO.AddServs)
        {
            oldAS[a.IdAdd] = a.Count;
            dataAS[dataAS.FindIndex(x => x.IdAdd ==
a.IdAdd)].Count = a.Count;
        }

        gridAServ.DataSource = new BindingSource(dataAS, null);
        gridAServ = TableFunc.AddHeader(gridAServ, new
List<string> { "ID", "Название", "Стоимость", "Устарела",

```

```

"Количество" });
    for (int i = 0; i < 4; i++)
    {
        gridAServ.Columns[i].ReadOnly = true;
    }
}

private void save_Click(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(school.Text) ||
string.IsNullOrEmpty(contact.Text) || !datetime.MaskFull ||
string.IsNullOrEmpty(meetpoint.Text) || serv.Items.Count ==
0 || price.Items.Count == 0 || guide.Items.Count == 0)
    {
        MessageBox.Show("Заполнены не все обязательные
поля");
    }
    else
    {
        dataO.School = school.Text;
        dataO.Contact = contact.Text;
        dataO.DateTime =
DateTime.ParseExact(datetime.Text, "dd.MM.yyyy HH:mm",
CultureInfo.InvariantCulture);
        dataO.MeetPoint = meetpoint.Text;
        dataO.PartyCount = (int)party.Value;
        dataO.AccomCount = (int)accom.Value;
        dataO.IdGroup = (int)age.SelectedValue;
        dataO.IdEmp =
((Employee)guide.SelectedItem).IdEmp;
        dataO.IdPrice =
((Price)price.SelectedItem).IdPrice;
        dataO.Cost = float.Parse(cost.Text);
        dataO.Prepay = float.Parse(prepay.Text);

        if (dataO.IdStat == 0)
        {
            dataO.IdStat = 1;
        }

        dataO.Current = CurrentUser.current.Login;

        if (change)
        {
            state = JsonFunc.Request("UpdateOrdr", "PUT",
JsonConvert.SerializeObject(dataO));
        }
        else
        {
            state = JsonFunc.Request("AddOrdr", "POST",
JsonConvert.SerializeObject(dataO));
        }
        if (state == "-2" || state == "-1")

```

```

        {
            MessageBox.Show("Ошибка доступа к данным.
Пожалуйста, обратитесь к администратору.");
        }

        if (!change)
        {
            dataO.IdOrdr = int.Parse(state);
        }

        foreach (DataGridViewRow row in gridAServ.Rows)
        {
            if ((int)row.Cells[4].Value > 0 &&
!oldAS.ContainsKey((int)row.Cells[0].Value))
            {
                state = JsonFunc.Request("AddAServOrdr",
"POST",
                JsonConvert.SerializeObject(new
AddServOrdr((int)row.Cells[0].Value,
                dataO.IdOrdr,
(int)row.Cells[4].Value));
            }
            else if ((int)row.Cells[4].Value == 0 &&
oldAS.ContainsKey((int)row.Cells[0].Value))
            {
                state
                =
                JsonFunc.Delete("DeleteAServOrdr", (int)row.Cells[0].Value + "/"
+ dataO.IdOrdr);
            }
            else if ((int)row.Cells[4].Value > 0 &&
oldAS[(int)row.Cells[0].Value] != (int)row.Cells[4].Value)
            {
                state
                =
                JsonFunc.Request("UpdateAServOrdr",
                "PUT",
                JsonConvert.SerializeObject(new
AddServOrdr((int)row.Cells[0].Value,
                dataO.IdOrdr,
(int)row.Cells[4].Value));
            }
            if (state == "-2" || state == "-1")
            {
                MessageBox.Show("Ошибка доступа к данным.
Пожалуйста, обратитесь к администратору.");
                break;
            }
        }

        Close();
    }

    private void text(object sender, KeyPressEventArgs e)
    {
        char number = e.KeyChar;
        if (!char.IsWhiteSpace(number) &&
!char.IsLetterOrDigit(number) && !char.IsPunctuation(number) &&

```

```

number != 8 && number != 46)
{
    e.Handled = true;
}
}
private void ComboBox_DropDown(object sender,
System.EventArgs e)
{
    ComboBox senderComboBox = (ComboBox)sender;
    int width = senderComboBox.DropDownWidth;
    Graphics g = senderComboBox.CreateGraphics();
    Font font = senderComboBox.Font;
    int vertScrollBarWidth =
        (senderComboBox.Items.Count >
senderComboBox.MaxDropDownItems)
        ? SystemInformation.VerticalScrollBarWidth : 0;

    int newWidth;
    foreach (object s in ((ComboBox)sender).Items)
    {
        newWidth = (int)g.MeasureString(s.ToString(),
font).Width
            + vertScrollBarWidth;
        if (width < newWidth)
        {
            width = newWidth;
        }
    }
    senderComboBox.DropDownWidth = width;
}

private void find_service()
{
    Service old = new Service();
    if (serv.Items.Count > 0)
    {
        old = (Service)serv.SelectedItem;
    }

    AvailableS = dataS.FindAll(x =>
x.Ages.Contains(age.SelectedItem) && !x.Outdated
    && x.Days.Exists(y => y.DayNum ==
(int)DateTime.ParseExact(datetime.Text, "dd.MM.yyyy HH:mm",
CultureInfo.InvariantCulture).DayOfWeek));
    serv.DataSource = new BindingSource(AvailableS, null);

    if (serv.Items.Count > 0 && AvailableS.Contains(old))
    {
        old = (Service)serv.SelectedItem;
    }

    find_price();
    find_emp();
}

```

```

    }
    private void find_price()
    {
        Price old = new Price();
        if (price.Items.Count > 0)
        {
            old = (Price)price.SelectedItem;
        }

        if (serv.Items.Count > 0)
        {
            AvailableP
            ((Service)serv.SelectedItem).Prices.FindAll(x => !x.Outdated &&
            x.MemberMin <= party.Value
            && x.MemberMax >= party.Value && x.AccomCount
            == accom.Value);
        }
        else
        {
            AvailableP = new List<Price>();
        }
        price.DataSource = new BindingSource(AvailableP, null);
        cost_changed();

        if (price.Items.Count > 0 && AvailableP.Contains(old))
        {
            old = (Price)price.SelectedItem;
        }
    }
    private void find_emp()
    {
        Employee old = new Employee();
        if (guide.Items.Count > 0)
        {
            old = (Employee)guide.SelectedItem;
        }

        if (serv.Items.Count > 0)
        {
            AvailableE = dataE.FindAll(x =>
            x.Guide.Contains((Service)serv.SelectedItem) &&
            !x.Orders.Exists(y => y.DateTime.Date ==
            DateTime.ParseExact(datetime.Text, "dd.MM.yyyy HH:mm",
            CultureInfo.InvariantCulture).Date && y.IdOrdr != dataO.IdOrdr));
        }
        guide.DataSource = new BindingSource(AvailableE, null);

        if (guide.Items.Count > 0 && AvailableE.Contains(old))
        {
            old = (Employee)guide.SelectedItem;
        }
    }
}

```



```

private void memb_ValueChanged(object sender, EventArgs e)
{
    find_price();
}

private void age_SelectedValueChanged(object sender,
EventArgs e)
{
    DateTime now;
    if (DateTime.TryParseExact(datetime.Text, "dd.MM.yyyy
HH:mm", CultureInfo.InvariantCulture, DateTimeStyles.None, out
now))
    {
        find_service();
    }
}

private void datetime_Validated(object sender, EventArgs e)
{
    find_service();
    find_emp();
}

private void price_SelectedIndexChanged(object sender,
EventArgs e)
{
    cost_changed();
}

private void cost_changed()
{
    if (price.Items.Count > 0)
    {
        currentCost = ((Price)price.SelectedItem).Cost;
    }
    else
    {
        currentCost = 0;
    }

    foreach (DataGridViewRow row in gridAServ.Rows)
    {
        if ((int)row.Cells["Count"].Value > 0)
        {
            currentCost +=
(double)row.Cells["Cost"].Value * (int)row.Cells["Count"].Value;
        }
    }
    cost.Text = currentCost.ToString();
    prepay.Text = (currentCost * 0.33).ToString();
}

private void gridAServ_CellValueChanged(object sender,

```

```

DataGridViewCellEventArgs e)
{
    cost_changed();
}
}

```

## Г.10 Окно отчетов

```

public partial class Reports : Form
{
    private Form parent;
    private bool backcommand;
    private DateTime bgn = new DateTime(0001, 1, 1);
    private DateTime nd = new DateTime(9999, 12, 31);
    private List<Age> ages;
    private List<Ordr> ordrs;
    private List<AgeList> a;
    private List<OrdrList> o;

    private class AgeList
    {
        public string Ages { get; set; }
        public int Count { get; set; }
    }

    private class OrdrList
    {
        public string Date { get; set; }
        public int Count { get; set; }
    }

    public Reports(Form p)
    {
        parent = p;
        backcommand = false;
        InitializeComponent();
    }

    private void Reports_Load(object sender, EventArgs e)
    {
        Load_data();
    }

    private void Reports_FormClosed(object sender,
    FormClosedEventArgs e)
    {
        if (!backcommand)
        {
            Application.Exit();
        }
    }
}

```

```

private void back_Click(object sender, EventArgs e)
{
    backcommand = true;
    Close();
    parent.Show();
}

private void excel_Click(object sender, EventArgs e)
{
    SaveFileDialog dialog = new SaveFileDialog();
    dialog.DefaultExt = "xlsx";
    dialog.Filter = "Книга Excel (*.xlsx) | *.xlsx";
    dialog.AddExtension = true;
    if (dialog.ShowDialog() == DialogResult.OK)
    {
        Ex.Application excelobj = new Ex.Application();
        excelobj.DisplayAlerts = false;
        excelobj.SheetsInNewWorkbook = 2;
        Ex.Workbook book = excelobj.Workbooks.Add();

        Ex.Worksheet exAges =
        (Ex.Worksheet)book.Worksheets.get_Item(1);
        exAges.Name = "KidStr - Возраста";
        exAges.Cells[1, 1] = "Возрастные категории";
        exAges.Cells[1, 2] = "Количество";

        exAges.Cells[1, 5] = "Начало:";
        exAges.Cells[1, 6] = begin.MaskCompleted ?
begin.Text : "-";
        exAges.Cells[1, 7] = "Конец:";
        exAges.Cells[1, 8] = end.MaskCompleted ? end.Text
: "-";

        for (int i = 0; i < a.Count; i++)
        {
            exAges.Cells[i + 2, 1] = a[i].Ages;
            exAges.Cells[i + 2, 2] = a[i].Count;
        }

        int k = a.Count + 1;
        Ex.Range excelcells = exAges.get_Range("A1", "B" +
k);
        excelcells.Borders.LineStyle =
Ex.XlLineStyle.xlContinuous;
        exAges.Columns.AutoFit();
        Ex.ChartObjects xlCharts =
        (Ex.ChartObjects)exAges.ChartObjects();
        Ex.ChartObject myChart = xlCharts.Add(200, 50, 300,
250);

        Ex.Chart chartPage = myChart.Chart;
        chartPage.SetSourceData(excelcells);
        chartPage.ChartType = Ex.XlChartType.xlPie;
        chartPage.HasTitle = false;
    }
}

```

```

        Ex.Worksheet                exOrdrs                =
        (Ex.Worksheet)book.Worksheets.get_Item(2);
        exOrdrs.Name = "KidStr - Заказы";
        exOrdrs.Cells[1, 1] = "Месяца";
        exOrdrs.Cells[1, 2] = "Количество";

        exOrdrs.Cells[1, 5] = "Начало:";
        exOrdrs.Cells[1, 6] = begin.MaskCompleted ?
begin.Text : "-";
        exOrdrs.Cells[1, 7] = "Конец:";
        exOrdrs.Cells[1, 8] = end.MaskCompleted ? end.Text
        : "-";

        for (int i = 0; i < o.Count; i++)
        {
            exOrdrs.Cells[i + 2, 1] = o[i].Date;
            exOrdrs.Cells[i + 2, 2] = o[i].Count;
        }
        k = o.Count + 1;
        excelcells = exOrdrs.get_Range("A1", "B" + k);
        excelcells.Borders.LineStyle                =
Ex.XlLineStyle.xlContinuous;
        exOrdrs.Columns.AutoFit();
        xlCharts                =
        (Ex.ChartObjects)exOrdrs.ChartObjects();
        myChart = xlCharts.Add(200, 50, 300, 250);
        chartPage = myChart.Chart;
        chartPage.SetSourceData(excelcells);
        chartPage.ChartType                =
Ex.XlChartType.xlColumnStacked;
        chartPage.HasTitle = false;
        chartPage.HasLegend = false;

        book.SaveAs(dialog.FileName);
        book.Close();
        excelobj.Quit();
    }
}

private void report_Click(object sender, EventArgs e)
{
    if (begin.MaskCompleted)
    {
        bgn                =                DateTime.ParseExact(begin.Text,
"dd.MM.yyyy", CultureInfo.InvariantCulture);
    }
    else
    {
        bgn = ordrs.Select(x => x.DateTime).Min();
    }
}

```

```

        if (end.MaskCompleted)
        {
            nd = DateTime.ParseExact(end.Text, "dd.MM.yyyy",
CultureInfo.InvariantCulture);
        }
        else
        {
            nd = orders.Select(x => x.DateTime).Max();
        }
        Load_data();
    }
    private void Load_data()
    {
        ages =
        JsonConvert.DeserializeObject<List<Age>>(JsonFunc.ReturnList("Ag
es"));
        orders =
        JsonConvert.DeserializeObject<List<Ordr>>(JsonFunc.ReturnList("O
rdrs"));
        if (orders.Count == 0)
        {
            MessageBox.Show("Данные для отчетов
отсутствуют.");
            Close();
        }
        foreach (Ordr ord in orders)
        {
            ord.Age = ages.First(x => x.IdGroup ==
ord.IdGroup);
        }

        if (bgn > nd)
        {
            MessageBox.Show("Недопустимый временной
промежуток.");
            return;
        }

        a = orders.Where(x => x.DateTime >= bgn && x.DateTime <=
nd).OrderBy(x => x.IdGroup).GroupBy(x => x.AgeName).Select(y =>
new AgeList { Ages = y.Key, Count = y.Count() }).ToList();
        o = orders.Where(x => x.DateTime >= bgn && x.DateTime <=
nd).OrderBy(x => x.DateTime).GroupBy(x =>
x.DateTime.ToString("MMM yyyy")).Select(y => new OrdrList { Date
= y.Key, Count = y.Count() }).ToList();

        diagAges.DataSource = new BindingSource(a, null);
        diagAges.Series[0].XValueMember = "Ages";
        diagAges.Series[0].YValueMembers = "Count";
        diagAges.DataBind();

        diagOrders.DataSource = new BindingSource(o, null);
        diagOrders.Series[0].XValueMember = "Date";
    }

```

```
diagOrdrs.Series[0].YValueMembers = "Count";  
diagOrdrs.Series[0].IsVisibleInLegend = false;  
diagOrdrs.DataBind();  
}  
}
```

## ПРИЛОЖЕНИЕ Д

### Результаты тестирования АИС

Таблица Д.1 – Тестирование АИС управления заказами на экскурсии

Test Title	Test Steps	Test Data	Expected Result	Actual Result	Status
1) Проверка возможности входа на панель менеджера при верных логине и пароле.	а) Ввести тестовые данные б) Нажать «ВОЙТИ».	Логин: Manager Пароль: Mprassword	Закрывается форма авторизации, открывается главная форма менеджера.	Закрывается форма авторизации, открывается главная форма менеджера.	Passed
2) Проверка возможности входа на панель администратора при верных логине и пароле.	а) Ввести тестовые данные. б) Нажать «ВОЙТИ».	Логин: Admin Пароль: Aprassword Войти как администратор: выбрано	Закрывается форма авторизации, открывается главная форма администратора.	Закрывается форма авторизации, открывается главная форма администратора.	Passed
3) Проверка возможности входа на панель администратора при отсутствии прав доступа.	а) Ввести тестовые данные. б) Нажать «ВОЙТИ».	Логин: Manager Пароль: Mprassword Войти как администратор: выбрано	Выводится сообщение «Нет прав доступа к панели администратора».	Выводится сообщение «Нет прав доступа к панели администратора».	Passed

Продолжение таблицы Д.1

Test Title	Test Steps	Test Data	Expected Result	Actual Result	Status
4) Проверка возможности входа в систему при неверной комбинации логина и пароля.	а) Ввести тестовые данные. б) Нажать «ВОЙТИ».	Логин: Manager Пароль: MM	Выводится сообщение «Неверный логин/пароль».	Выводится сообщение «Неверный логин/пароль».	Passed
5) Проверка возможности добавить мероприятие.	а) После теста 1 нажать «Мероприятия». б) Нажать «Добавить». в) Ввести тестовые данные. г) Нажать «Сохранить».	Название: Мероприятие Тип: Экскурсия Направление: Производство Продолжительность: 2-3 часа Описание: Описание Условия: Условия Возраста: 7 класс: Выбрать Доступно: ПН: Выбрать	Окно закрывается, в конце списка мероприятий добавляется новая запись с введенными данными.	Окно закрывается, в конце списка мероприятий добавляется новая запись с введенными данными.	Passed
6) Проверка возможности добавить мероприятие при незаполненных полях.	а) После теста 1 нажать «Мероприятия». б) Нажать «Добавить». в) Нажать «Сохранить».		Выводится сообщение «Заполнены не все обязательные поля».	Выводится сообщение «Заполнены не все обязательные поля».	Passed
7) Проверка возможности изменить мероприятие.	а) После теста 5 нажать на последнее мероприятие в списке. б) Нажать «Изменить». в) Ввести тестовые данные. г) Нажать «Сохранить».	Название: Мероприятие2 Тип: Экскурсия Направление: Производство Продолжительность: 3-4 часа Описание: Описание2 Условия: Условия2 Возраста: 8 класс: Выбрать Доступно: ВТ: Выбрать	Окно закрывается, у изменяемой записи обновляются все данные в соответствии с введенными.	Окно закрывается, у изменяемой записи обновляются все данные в соответствии с введенными.	Passed



Продолжение таблицы Д.1

Test Title	Test Steps	Test Data	Expected Result	Actual Result	Status
8) Проверка возможности удалить мероприятие.	а) После теста 7 нажать на последнее мероприятие в списке. б) Нажать «Удалить». в) Нажать «Да».		Запись удаляется из списка.	Запись удаляется из списка.	Passed
9) Проверка возможности добавить цену мероприятия.	а) После теста 5 нажать на последнее мероприятие в списке. б) Нажать «Изменить». в) Нажать «Добавить». г) Внести тестовые данные. д) Нажать «Сохранить». е) Нажать «Сохранить». ж) Нажать «Детали».	Минимум: 1 Максимум: 10 Сопровождающих: 1 Стоимость: 10000 Условия: Условия	В конец списка цен добавляется новая запись с введенными данными.	В конец списка цен добавляется новая запись с введенными данными.	Passed
10) Проверка возможности добавить цену при мероприятии незаполненных полях.	а) После теста 5 нажать на последнее мероприятие в списке. б) Нажать «Изменить». в) Нажать «Добавить». г) Внести тестовые данные. д) Нажать «Сохранить».		Выводится сообщение «Заполнены не все обязательные поля».	Выводится сообщение «Заполнены не все обязательные поля».	Passed

Продолжение таблицы Д.1

Test Title	Test Steps	Test Data	Expected Result	Actual Result	Status
11) Проверка возможности изменить цену мероприятия.	а) После теста 5 нажать на последнее мероприятие в списке. б) Нажать «Изменить». в) Нажать «Изменить». г) Внести тестовые данные. д) Нажать «Сохранить». е) Нажать «Сохранить». ж) Нажать «Детали».	Минимум: 2 Максимум: 11 Сопровождающих: 2 Стоимость: 11000 Условия: Условия2	Запись по первой цене в списке цен содержит внесенные значения.	Запись по первой цене в списке цен содержит внесенные значения.	Passed
12) Проверка возможности удалить цену мероприятия.	а) После теста 5 нажать на последнее мероприятие в списке. б) Нажать «Изменить». в) Нажать «Удалить». г) Нажать «Да». д) Нажать «Сохранить». е) Нажать «Детали».		Первая цена в списке цен удаляется из списка.	Первая цена в списке цен удаляется из списка.	Passed
13) Проверка возможности удалить мероприятие, входящее в заказ.	а) После теста 1 нажать «Мероприятия» б) Нажать на первое мероприятие в списке. в) Нажать «Удалить». г) Нажать «Да».		У удаляемой записи в списке отмечается флажком поле «Устарело».	У удаляемой записи в списке отмечается флажком поле «Устарело».	Passed

## Продолжение таблицы Д.1

Test Title	Test Steps	Test Data	Expected Result	Actual Result	Status
14) Проверка возможности добавить доп. услугу	а) После теста 1 нажать «Доп. услуги». б) Нажать «Добавить». в) Ввести тестовые данные. г) Нажать «Сохранить».	Название: Услуга Цена: 1000	Окно закрывается, в конец списка доп. услуг добавляется новая запись с введенными данными.	Окно закрывается, в конец списка доп. услуг добавляется новая запись с введенными данными.	Passed
15) Проверка возможности добавить доп. услугу при незаполненных полях.	а) После теста 1 нажать «Доп. услуги» б) Нажать «Добавить». в) Нажать «Сохранить».		Выводится сообщение «Заполнены не все обязательные поля».	Выводится сообщение «Заполнены не все обязательные поля».	Passed
16) Проверка возможности изменить доп. услугу.	г) После теста 14 нажать на последнюю доп. услугу в списке. д) Нажать «Изменить». е) Ввести тестовые данные. ж) Нажать «Сохранить».	Название: Услуга2 Цена: 1200	Окно закрывается, у изменяемой записи обновляются все данные в соответствии с введенными.	Окно закрывается, у изменяемой записи обновляются все данные в соответствии с введенными.	Passed
17) Проверка возможности удалить доп. услугу.	а) После теста 16 нажать на последнюю доп. услугу в списке. б) Нажать «Удалить». в) Нажать «Да».		Запись удаляется из списка.	Запись удаляется из списка.	Passed
18) Проверка возможности удалить доп. услугу, входящую в заказ.	а) После теста 1 нажать «Доп. услуги» б) Нажать на первую доп. услугу в списке. в) Нажать «Удалить». г) Нажать «Да».		У удаляемой записи в списке отмечается флажком поле «Устарела».	У удаляемой записи в списке отмечается флажком поле «Устарела».	Passed

## Продолжение таблицы Д.1

Test Title	Test Steps	Test Data	Expected Result	Actual Result	Status
19) Проверка возможности добавить заказ.	а) После теста 1 нажать «Заказы». б) Нажать «Добавить». в) Ввести тестовые данные. г) Нажать «Сохранить».	Организация: Организация Возраст/Класс: 5 лет Контактное лицо: Лицо Дата и время встречи: 11.06.2022 13:00 Точка встречи: Точка Экскурсовод: 3333 333333   Name3 Услуга: 2   Мастер-класс1 Вид цены: 11-15 чел.   1 сопр.   31500 р Участников: 10 Сопровождающих: 1	Окно закрывается, в конец списка заказов добавляется новая запись с введенными данными.	Окно закрывается, в конец списка заказов добавляется новая запись с введенными данными.	Passed
20) Проверка возможности добавить заказ при незаполненных полях.	а) После теста 1 нажать «Заказы» б) Нажать «Добавить». в) Нажать «Сохранить».		Выводится сообщение «Заполнены не все обязательные поля».	Выводится сообщение «Заполнены не все обязательные поля».	Passed
21) Проверка возможности изменить заказ.	а) После теста 19 нажать на последний заказ в списке. б) Нажать «Изменить». в) Ввести тестовые данные. г) Нажать «Сохранить».	Организация: Организация2 Возраст/Класс: 6 лет Контактное лицо: Лицо2 Дата и время встречи: 11.06.2022 14:00 Точка встречи: Точка2 Экскурсовод: 3333 333333   Name3 Услуга: 2   Мастер-класс1 Вид цены: 16-20 чел.   2 сопр.   38500р Участников: 17 Сопровождающих: 2	Окно закрывается, у изменяемой записи обновляются все данные в соответствии с введенными.	Окно закрывается, у изменяемой записи обновляются все данные в соответствии с введенными.	Passed

# Продолжение таблицы Д.1

Test Title	Test Steps	Test Data	Expected Result	Actual Result	Status
22) Проверка возможности удалить заказ.	а) После теста 21 нажать на последний заказ в списке. б) Нажать «Удалить». в) Нажать «Да».		Запись удаляется из списка.	Запись удаляется из списка.	Passed
23) Проверка возможности добавить доп. услугу в заказ.	а) После теста 21 нажать на последний заказ в списке. б) Нажать «Изменить». в) Ввести тестовые данные. г) Нажать «Сохранить». д) Нажать «Детали».	Доп. услуги: Фотограф: 1	В списке доп. услуг для выбранной доп. услуги сохраняются введенные значения.	В списке доп. услуг для выбранной доп. услуги сохраняются введенные значения.	Passed
24) Проверка возможности изменить доп. услугу в заказе.	а) После теста 23 нажать на последний заказ в списке. б) Нажать «Изменить». в) Ввести тестовые данные. г) Нажать «Сохранить». д) Нажать «Детали».	Доп. услуги: Фотограф: 2	В списке доп. услуг для выбранной доп. услуги сохраняются введенные значения.	В списке доп. услуг для выбранной доп. услуги сохраняются введенные значения.	Passed
25) Проверка возможности удалить доп. услугу из заказа.	а) После теста 24 нажать на последний заказ в списке. б) Нажать «Изменить». в) Ввести тестовые данные. г) Нажать «Сохранить». д) Нажать «Детали».	Доп. услуги: Фотограф: 0	В списке доп. услуг удаленная доп. услуга отсутствует.	В списке доп. услуг удаленная доп. услуга отсутствует.	Passed

## Продолжение таблицы Д.1

Test Title	Test Steps	Test Data	Expected Result	Actual Result	Status
26) Проверка возможности изменить заказ с предоплатой.	а) После теста 1 нажать «Заказы». б) Нажать на первый заказ в списке. в) Нажать «Изменить». г) Нажать «Да».		Выводится сообщение «Заказ уже предоплачен».	Выводится сообщение «Заказ уже предоплачен».	Passed
27) Проверка возможности удалить заказ с предоплатой.	а) После теста 1 нажать «Заказы». б) Нажать на первый заказ в списке. в) Нажать «Удалить». г) Нажать «Да».		Выводится сообщение «Заказ уже предоплачен».	Выводится сообщение «Заказ уже предоплачен».	Passed
28) Проверка возможности изменить статус заказа.	а) После теста 21 нажать на последний заказ в списке. б) Нажать «Статус». в) Внести тестовые данные. г) Нажать «Ок».	Статус: «Оплачен»	Окно закрывается, у изменяемой записи обновляется статус.	Окно закрывается, у изменяемой записи обновляется статус.	Passed
29) Проверка возможности добавить тип мероприятия.	а) После теста 2 нажать «Параметры мероприятий». б) Нажать «Типы». в) Нажать «Добавить». г) Ввести тестовые данные. д) Нажать «Сохранить».	Название: Тип	Окно закрывается, в конце списка типов добавляется новая запись с введенными данными.	Окно закрывается, в конце списка типов добавляется новая запись с введенными данными.	Passed

Продолжение таблицы Д.1

Test Title	Test Steps	Test Data	Expected Result	Actual Result	Status
30) Проверка возможности добавить тип мероприятия при незаполненных полях.	а) После теста 2 нажать «Параметры мероприятий». б) Нажать «Типы». в) Нажать «Добавить». г) Нажать «Сохранить».		Выводится сообщение «Заполнены не все обязательные поля».	Выводится сообщение «Заполнены не все обязательные поля».	Passed
31) Проверка возможности изменить тип мероприятия.	а) После теста 29 нажать на последний тип в списке. б) Нажать «Изменить». в) Ввести тестовые данные. г) Нажать «Сохранить».	Название: Тип2	Окно закрывается, у изменяемой записи обновляются все данные в соответствии с введенными.	Окно закрывается, у изменяемой записи обновляются все данные в соответствии с введенными.	Passed
32) Проверка возможности удалить тип мероприятия.	а) После теста 31 нажать на последний тип в списке. б) Нажать «Удалить». в) Нажать «Да».		Запись удаляется из списка.	Запись удаляется из списка.	Passed
33) Проверка возможности удалить тип, включающий мероприятия.	а) После теста 2 нажать «Параметры мероприятий». б) Нажать «Типы». в) Нажать на первый тип в списке. г) Нажать «Удалить». д) Нажать «Да».		У удаляемой записи в списке отмечается флажком поле «Устарел».	У удаляемой записи в списке отмечается флажком поле «Устарел».	Passed

## Продолжение таблицы Д.1

Test Title	Test Steps	Test Data	Expected Result	Actual Result	Status
34) Проверка возможности добавления направлений мероприятия.	а) После теста 2 нажать «Параметры мероприятия». б) Нажать «Направления». в) Нажать «Добавить». г) Ввести тестовые данные. д) Нажать «Сохранить».	Название: Направление	Окно закрывается, в конце списка направлений добавляется новая запись с введенными данными.	Окно закрывается, в конце списка направлений добавляется новая запись с введенными данными.	Passed
35) Проверка возможности добавления направлений мероприятия при незаполненных полях.	а) После теста 2 нажать «Параметры мероприятия». б) Нажать «Направления». в) Нажать «Добавить». г) Нажать «Сохранить».		Выводится сообщение «Заполнены не все обязательные поля».	Выводится сообщение «Заполнены не все обязательные поля».	Passed
36) Проверка возможности изменения направления мероприятия.	а) После теста 34 нажать на последнее направление в списке. б) Нажать «Изменить». в) Ввести тестовые данные. г) Нажать «Сохранить».	Название: Направление2	Окно закрывается, у изменяемой записи обновляются все данные в соответствии с введенными.	Окно закрывается, у изменяемой записи обновляются все данные в соответствии с введенными.	Passed
37) Проверка возможности удаления направления мероприятия.	а) После теста 36 нажать на последнее направление в списке. б) Нажать «Удалить». в) Нажать «Да».		Запись удаляется из списка.	Запись удаляется из списка.	Passed



# Продолжение таблицы Д.1

Test Title	Test Steps	Test Data	Expected Result	Actual Result	Status
38) Проверка возможности удалить направление, включающее мероприятия.	<p>а) После теста 2 нажать «Параметры мероприятий».</p> <p>б) Нажать «Направления».</p> <p>в) Нажать на первый тип в списке.</p> <p>г) Нажать «Удалить».</p> <p>д) Нажать «Да».</p>		У удаляемой записи в списке отмечается флажком поле «Устарело».	У удаляемой записи в списке отмечается флажком поле «Устарело».	Passed
39) Проверка возможности добавить сотрудника.	<p>а) После теста 2 нажать «Сотрудники».</p> <p>б) Нажать «Добавить».</p> <p>в) Ввести тестовые данные.</p> <p>г) Нажать «Сохранить».</p>	<p>Паспорт: 8888 888888</p> <p>ФИО: Name8</p> <p>Телефон: +7(988)8888888</p>	Окно закрывается, в конце списка сотрудников добавляется новая запись с введенными данными.	Окно закрывается, в конце списка сотрудников добавляется новая запись с введенными данными.	Passed
40) Проверка возможности добавить при незаполненных полях.	<p>а) После теста 2 нажать «Сотрудники».</p> <p>б) Нажать «Добавить».</p> <p>в) Нажать «Сохранить».</p>		Выводится сообщение «Заполнены не все обязательные поля».	Выводится сообщение «Заполнены не все обязательные поля».	Passed
41) Проверка возможности изменить данные сотрудника.	<p>а) После теста 39 нажать на последнего сотрудника в списке.</p> <p>б) Нажать «Изменить».</p> <p>в) Ввести тестовые данные.</p> <p>г) Нажать «Сохранить».</p>	<p>Паспорт: 8888 888889</p> <p>ФИО: Name89</p> <p>Телефон: +7(988)8888889</p>	Окно закрывается, у изменяемой записи обновляются все данные в соответствии с введенными.	Окно закрывается, у изменяемой записи обновляются все данные в соответствии с введенными.	Passed

# Продолжение таблицы Д.1

Test Title	Test Steps	Test Data	Expected Result	Actual Result	Status
42) Проверка возможности удалить данные сотрудника.	а) После теста 41 нажать на последнего сотрудника в списке. б) Нажать «Удалить». в) Нажать «Да»		Запись удаляется из списка.	Запись удаляется из списка.	Passed
43) Проверка возможности добавить мероприятие сотруднику.	а) После теста 39 нажать на последнего сотрудника в списке. б) Нажать «Изменить». в) Ввести тестовые данные. г) Нажать «Сохранить». д) Нажать «Детали».	Может вести следующие мероприятия: Мастер-Класс1: Выбрано: Отмечено	Выбранное мероприятие присутствует в списке мероприятий сотрудника.	Выбранное мероприятие присутствует в списке мероприятий сотрудника.	Passed
44) Проверка возможности удалить мероприятие у сотрудника.	а) После теста 39 нажать на последнего сотрудника в списке. б) Нажать «Изменить». в) Ввести тестовые данные. г) Нажать «Сохранить». д) Нажать «Детали».	Может вести следующие мероприятия: Мастер-Класс1: Выбрано: Не Отмечено	Выбранное мероприятие отсутствует в списке мероприятий сотрудника.	Выбранное мероприятие отсутствует в списке мероприятий сотрудника.	Passed
45) Проверка возможности отобразить отчеты за период.	а) После теста 2 нажать «Отчеты». б) Внести тестовые данные. в) Нажать «Обновить».	Начало: 10.01.2021 Конец: 10.06.2022	На форме отображаются круговая и столбчатая диаграммы.	На форме отображаются круговая и столбчатая диаграммы.	Passed
46) Проверка возможности отобразить отчеты за период при некорректном периоде.	а) После теста 2 нажать «Отчеты». б) Внести тестовые данные. в) Нажать «Обновить».	Начало: 10.06.2022 Конец: 10.01.2021	Выводится сообщение «Недопустимый временной промежуток».	Выводится сообщение «Недопустимый временной промежуток».	Passed

Продолжение таблицы Д.1

Test Title	Test Steps	Test Data	Expected Result	Actual Result	Status
47) Проверка возможности сохранить отчеты за период в Excel.	а) После теста 45 нажать «В Excel». б) Выбрать директорию «Загрузки». в) Сохранить файл.		В директории «Загрузки» появляется файл «Отчет. xlsx», содержащий круговую и столбчатую диаграммы.	В директории «Загрузки» появляется файл «Отчет. xlsx», содержащий круговую и столбчатую диаграммы.	Passed
48) Проверка возможности выхода из системы.	г) После теста 1 нажать «Выйти».		Окно закрывается, открывается форма авторизации с незаполненными полями.	Окно закрывается, открывается форма с незаполненными полями.	Passed
49) Проверка возможности возвращения в прошлое окно.	а) После теста 1 нажать «Мероприятия». б) Нажать «<».		Окно закрывается, открывается главная форма менеджера.	Окно закрывается, открывается главная форма менеджера.	Passed
50) Проверка возможности выхода из приложения из окна, отличного от окна «Авторизация».	а) После теста 1 нажать «Мероприятия». б) Нажать «X».		Приложение закрывается.	Приложение закрывается.	Passed