

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
Политехнический институт

Факультет вычислительной техники
Кафедра «Вышей и прикладной математики»

КУРСОВАЯ РАБОТА

по дисциплине «Проектирование программного обеспечения»
на тему «Автоматизированная система учета услуг автотранспортного
предприятия»

Выполнил: студент группы 16ВБ1

Дужников Н. С.

Проверила: к.т.н. доцент Черушева Т.В.

Работа защищена с оценкой

Пенза, 2020

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
Федеральное государственное бюджетное образовательное
учреждение высшего профессионального образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Политехнический институт

Факультет вычислительной техники

Кафедра «Вышей и прикладной математики»

«УТВЕРЖДАЮ»

Заведующий кафедрой

И. В. Бойков

(подпись, инициалы, фамилия)

« » 201 г.

ЗАДАНИЕ
на курсовую работу (проект)

по дисциплине «Проектирование программного обеспечения»

студенту группы 16ВБ1 Дужникову Никите Сергеевичу

ТЕМА РАБОТЫ: Автоматизированная система учета услуг
автотранспортного предприятия

ИСХОДНЫЕ ДАННЫЕ: Требуется спроектировать логическую модель
реляционной базы данных учета услуг автотранспортного предприятия,
реализовать ее в приложении с помощью программы «Microsoft Visual Studio
C# 2010». Список ключевых слов для построения логической модели базы
данных: код клиента, наименование, телефон, адрес, код ТС, марка, гос.
номер, тип ТС, код услуги, дата заказа, дата прибытия, город отправки, город
прибытия, расстояние, сумма.

Курсовая работа должна раскрывать тему; объём работы – 25–30 страниц (1,5 интервала 14 шрифт Times New Roman с соблюдением соответствующего стандарта вуза СТО ПГУ 3.12–2016).

СТРУКТУРА РАБОТЫ:

Введение: актуальность, практическая значимость, цель, задачи исследования, объект исследования.

Основная часть:

Первая глава имеет теоретический характер и раскрывает понятийно-категориальный аппарат, описывает постановку задачи, содержит схемы данных.

Вторая глава имеет исследовательский характер: требуется разработать программное обеспечение для реализации определённой структуры в среде «Microsoft Visual Studio C# 2010», описание принципов работы приложения, инструкция пользователю.

Заключение содержит выводы по всей работе.

Список литературы оформляется в соответствии с ГОСТ 7.0.5–2008.

Сроки выполнения с «11» февраля 2019 г. по «1» июня 2019 г.

Руководитель проекта к.т.н., доцент Т.В.Черушева
(должность) (подпись, инициалы, фамилия)

Задание получил: _____ Н. С. Дужников
(подпись) (инициалы, фамилия студента)

Пояснительная записка содержит 48 листов формата А4, 37 рисунков, 3 таблицы, 3 источника, 1 приложение.

ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ,
ЖИЗНЕННЫЙ ЦИКЛ, МОДЕЛИ, ТАБЛИЦА, СХЕМА ДАННЫХ, С#.

Цель работы – разработка программного обеспечения автоматизированной системы учета услуг автотранспортного предприятия и реализация приложения с помощью программ Microsoft Visual Studio С# 2010.

					ПензГУ 010304–5КР141.03ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата	Автоматизированная система учета услуг автотранспортного предприятия Пояснительная записка	Лит.	Лист	Листов
Разраб.		Дужников Н. С.					4	50
Разраб.								
Провер.		Черушева Т.В.				Группа 16ВБ1 4		
Н. Контр.								
Утверд.								

ВВЕДЕНИЕ.....	6
1. ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ	7
1.1. ПОСТАНОВКА ЗАДАЧИ	7
1.2. ЖИЗНЕННЫЙ ЦИКЛ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ.....	8
1.3. РАЗРАБОТКА БАЗЫ ДАННЫХ	11
1.4 МЕТОД ФУНКЦИОНАЛЬНОГО МОДЕЛИРОВАНИЯ SADT	13
1.5 МОДЕЛИРОВАНИЕ ПРОЦЕССОВ IDEF3	14
1.6. ИНФОРМАЦИОННАЯ МОДЕЛЬ	15
1.7 МОДЕЛЬ СОСТОЯНИЙ	16
1.8. МЕТОДЫ ПРОЕКТИРОВАНИЯ АРХИТЕКТУРЫ ПО	18
1.8.1. ДИАГРАММА КЛАССОВ.....	19
1.8.2. ДИАГРАММА ВАРИАНТОВ ИСПОЛЬЗОВАНИЯ	19
1.8.3. ДИАГРАММА ПОСЛЕДОВАТЕЛЬНОСТИ	20
1.8.4. КООПЕРАТИВНАЯ ДИАГРАММА.....	20
1.8.5. ДИАГРАММА ДЕЯТЕЛЬНОСТИ	21
2. ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ.....	22
2.1. ИНСТРУКЦИЯ ПОЛЬЗОВАТЕЛЮ	22
1. ЗАКЛАДКА «КЛИЕНТЫ И ТС».....	24
2. ЗАКЛАДКА «ЗАКАЗ УСЛУГИ».....	30
3. ЗАКЛАДКА «ОТЧЕТЫ»	33
ЗАКЛЮЧЕНИЕ	37
ПРИЛОЖЕНИЕ.....	38
СПИСОК ЛИТЕРАТУРЫ	50

ВВЕДЕНИЕ

Проектирование экономических информационных систем (ЭИС) – логически сложная, трудоёмкая и длительная работа, требующая высокой квалификации разработчиков. В процессе создания и функционирования ЭИС информационные потребности пользователей меняются, уточняются, что усложняет разработку и сопровождение таких систем.

Основная доля затрат приходится на прикладное программное обеспечение (ПО) и разработку базы данных (БД).

Необходимость контроля процесса разработки программного обеспечения привела к появлению совокупности методов и средств создания ПО, объединённых общим названием «программная инженерия». В основе её заложена идея: проектирование ПО есть формальный процесс, который можно изучать и совершенствовать.

Для успешной реализации проекта объект проектирования должен быть описан при помощи полных и непротиворечивых моделей архитектуры ПО. Здесь закладываются структурные элементы системы, связи между ними, иерархия подсистем.

Модель – это полное описание системы ПО с некоторой точки зрения. Моделирование является центральным звеном всей работы по созданию качественного ПО. Модели строятся для того, чтобы понять структуру и поведение создаваемой системы, облегчить управление процессом её создания, уменьшить возможный риск и документировать принимаемые проектные решения.

Язык моделирования должен включать элементы модели (фундаментальные концепции моделирования и их семантику), нотацию (визуальное представление элементов моделирования), руководство по использованию.

Конечная цель разработки ПО – получение работающих приложений (кода).

1. ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

1.1. Постановка задачи

Необходимо разработать программное обеспечение для реализации автоматизированной системы учета услуг автотранспортного предприятия.

Требования к приложению:

- 1) Выбор пользовательских функций – Закладки;
- 2) Вариант диаграммы – Столбиковая;
- 3) Каталоги с файлами БД не определяются в программе;

1.2. Жизненный цикл программного обеспечения

Понятие жизненного цикла ПО (ЖЦ ПО) является одним из базовых понятий программной инженерии.

ЖЦ ПО — это период времени, который начинается с момента решения о необходимости создания ПО и заканчивается полным изъятием его из эксплуатации.

Основным нормативным документом, регламентирующим состав процессов ЖЦ ПО, является международный стандарт ISO/IEC 12207: 1995 «Information Technology - Software Life Cycle Processes». Он определяет структуру ЖЦ, содержащую процессы, действия и задачи, которые должны быть выполнены во время создания ПО (его российский аналог ГОСТ Р ИСО/МЭК 12207-99 введен в действие в июле 2000 г.). В данном стандарте процесс определяется как совокупность взаимосвязанных действий, преобразующих некоторые входные данные в выходные. Каждый процесс характеризуется определенными задачами и методами их решения, исходными данными, полученными от других процессов, и результатами.

Каждый процесс разделен на набор действий, каждое действие — на набор задач. Каждый процесс, действие или задача инициируется и выполняется другим процессом по мере необходимости, причем не существует заранее определенных последовательностей выполнения (естественно, при сохранении связей по входным данным).

Модель жизненного цикла программного обеспечения — структура, содержащая процессы действия и задачи, которые осуществляются в ходе разработки, использования и сопровождения программного продукта. При выборе схемы модели жизненного цикла для конкретной предметной области, решаются вопросы включения важных для создаваемого продукта видов работ или не включения несущественных работ. К широко используемым типам моделей ЖЦ относятся следующие: каскадная, спиральная, инкрементная, эволюционная, стандартизованная и др.

В данной работе выбрана **каскадная модель**. Согласно данной модели ЖЦ работы и задачи процесса разработки обычно выполняются последовательно. Однако вспомогательные и организационные процессы обычно выполняются параллельно с процессом разработки. В данной модели возвращение к начальному процессу предусматривается после сопровождения и исправления ошибок.

Особенность такой модели состоит в фиксации последовательных процессов разработки программного продукта. В ее основу положена модель фабрики, где продукт проходит стадии от замысла до производства, затем передается заказчику как готовое изделие, изменение которого не предусмотрено, хотя возможна замена на другое подобное изделие в случае рекламации или некоторых ее деталей, вышедших из строя.

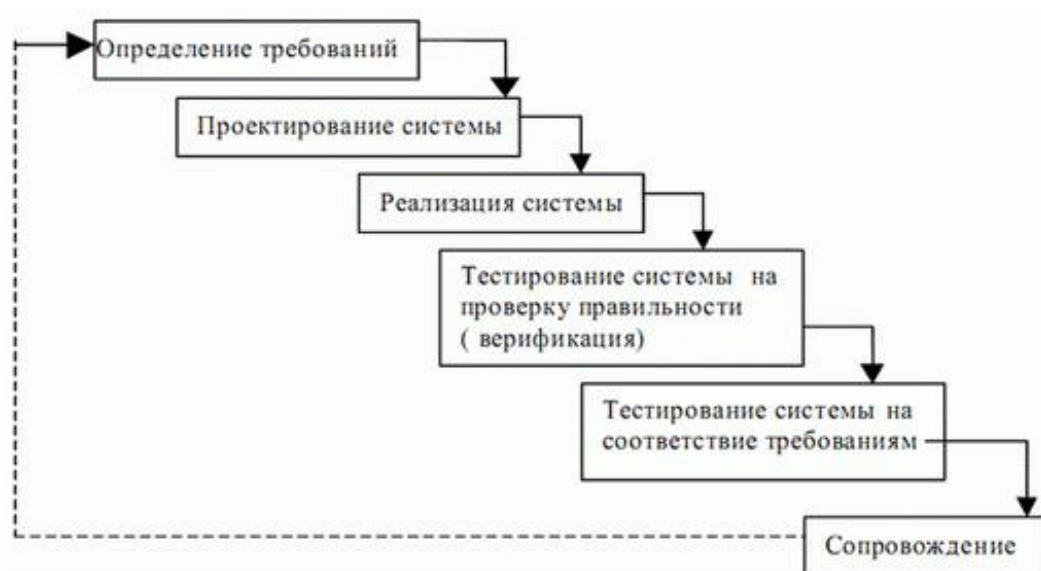


Рис 1. Каскадная модель ЖЦ

Недостатки этой модели:

- процесс создания ПС не всегда укладывается в такую жесткую форму и последовательность действий;
- не учитываются изменившиеся потребности пользователей, изменения во внешней среде, которые вызовут изменения требований к системе в ходе ее разработки;

- большой разрыв между временем внесения ошибки (например, на этапе проектирования) и временем ее обнаружения (при сопровождении), что приводит к большой переделке ПС.

При применении каскадной модели имеют место следующие факторы риска:

- требования к ПС недостаточно четко сформулированы, либо не учитывают перспективы развития ОС, сред и т.п.;
- большая система, не допускающая компонентной декомпозиции, может вызвать проблемы с размещением ее в памяти или на платформах, не предусмотренных в требованиях;
- внесение быстрых изменений в технологию и в требования может ухудшить процесс разработки отдельных частей системы или системы в целом;
- ограничения на ресурсы (человеческие, программные, технические и др.) в ходе разработки могут сузить отдельные возможности реализации системы;

Полученный продукт может оказаться плохим для применения по причине недопонимания разработчиками требований или функций системы или недостаточно проведенного тестирования.

Преимущества реализации системы с помощью каскадной модели следующие:

- все задачи подсистем и системы реализуются одновременно (ни одна задача не забыта), что способствует установлению стабильных связей и отношений между ними;
- полностью разработанную систему с документацией на нее легче сопровождать, тестировать, фиксировать ошибки и вносить изменения не беспорядочно, а целенаправленно, начиная с требований (например, добавить или заменять некоторые функции) и повторить процесс.

Разработанное ПО основано на каскадной модели. Это обосновано тем, что каждая работа выполняется полностью, и после ее завершения и перехода к следующему этапу возвращение к предыдущему не требуется. Промежуточный результат проверяется известными методами верификации и фиксируется в качестве готового эталона для следующего процесса

1.3. Разработка базы данных

Методика определения и документирования требований к базе данных заключается в составлении словаря данных. Словарь данных перечисляет и определяет отдельные элементы данных, которые должны храниться в базе (табл. 1).

Таблица 1

Элементы данных	Описание
Код клиента	Описание клиента
Наименование	
Адрес	
Телефон	
Марка	Описание транспортного средства
Гос. номер	
Тип. ТС	
Тип услуги	Грузовая или пассажирская
Дата заказа	Информация об услуге
Дата прибытия	
Город отправки	
Город прибытия	
Расстояние	
Сумма	

Далее модель развивается путем определения атрибутов для каждого объекта. Для этого из составленного ранее словаря данных выделяем необходимые элементы (табл.2).

Таблица 2. Объекты и атрибуты

	Объекты		
Атрибуты	Клиенты	Услуги	ТС
	Наименование	Тип услуги	Марка
	Адрес	Дата заказа	Гос. номер
	Телефон	Дата прибытия	Тип тс
		Город отправки	
		Город прибытия	
		Расстояние	
		Сумма	

Для того чтобы данные схемы стала реляционной моделью необходимо использование ключей (первичных и внешних) и отношений (рис.2).



Рис.2. Логическая модель

1.4 Метод функционального моделирования SADT

Метод функционального моделирования SADT представляет собой совокупность правил и процедур, предназначенных для построения функциональной модели объекта какой – либо предметной области. Функциональная модель SADT отображает функциональную структуру объекта, т.е. производимые им действия и связи между этими действиями.

Результатом применения метода SADT является модель, которая состоит из диаграмм, фрагментов текста и глоссария, имеющих ссылки друг на друга. Диаграммы – главные компоненты модели, все функции организации и интерфейсы на них представлены как блоки и дуги соответственно. Место соединения дуги с блоком определяет тип интерфейса. Управляющая информация входит в блок сверху, в то время как входная информация, которая подвергается обработке, показана с левой стороны блока, а результаты- с правой. Механизм, который осуществляет операцию, представляется дугой, входящей в блок снизу (рис.3).

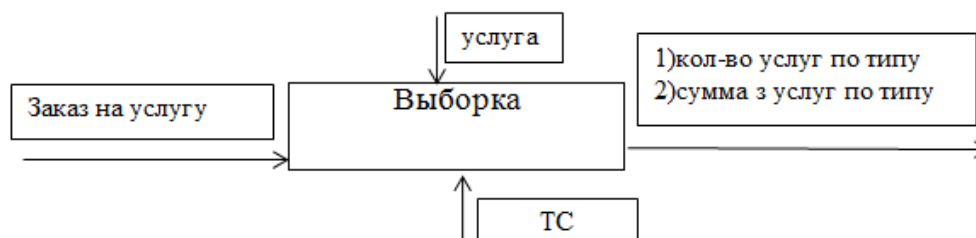


Рис.3.Функциональный блок и интерфейсные дуги

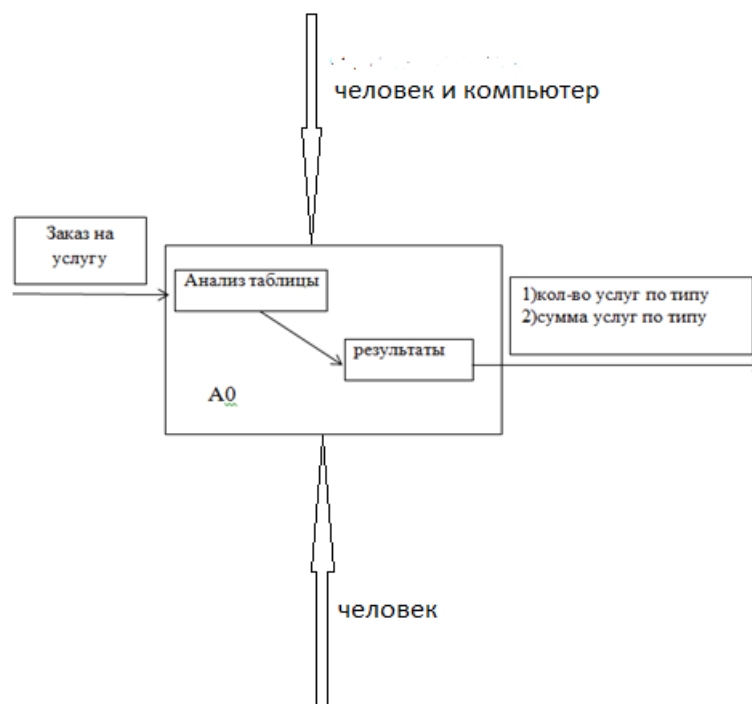


Рис. 4. Структура SADT-модели

- На вход поступает заявка на выполнение услуги (левая стрелка)
- На выходе выводится информация о количестве услуг по типу и о сумме данных услуг (правая стрелка)
- Механизм – это человек, оформляет услугу (нижняя стрелка)
- Управление – это человек и компьютер, выполняют подсчёт суммы (верхняя стрелка)

1.5 Моделирование процессов IDEF3

Основой модели IDEF3 служит сценарий процесса, который выделяет последовательность действий и подпроцессов анализируемой системы.

Взаимодействия между объектами изображаются с помощью связей, причем в IDEF3 они являются однонаправленными. Соединения (соединение «и», «исключающее или», «или») разбивают или соединяют внутренние потоки и используются для отображения ветвления процесса.

Для проведения количественного анализа диаграмм IDEF3 используются следующие показатели:

N – количество блоков на диаграмме

L – уровень декомпозиции

A_i – число стрелок, соединяющихся с i -м блоком диаграммы

А количественная оценка сбалансированности диаграммы может быть выполнена с помощью коэффициента сбалансированности:

$$K_b = \left| \sum A_i / N - \max A_i \right|$$

При этом необходимо стремиться к тому, чтобы значение K_b было минимальным.

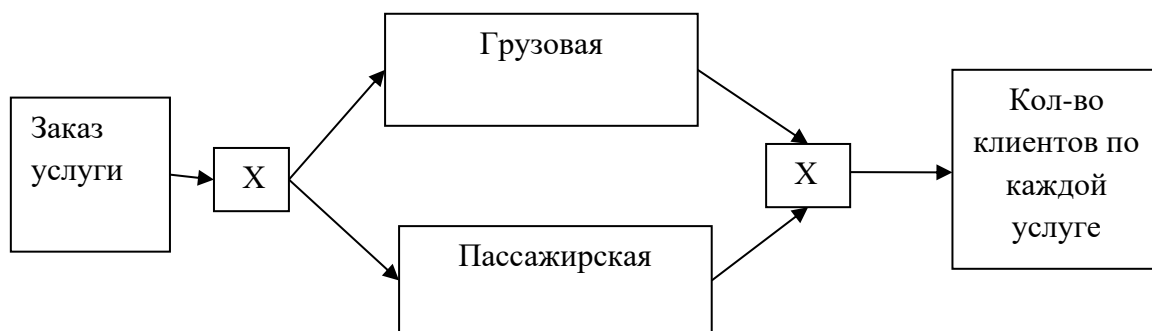


Рис.5. Модель IDEF3 разрабатываемого ПО

$$K_b = \left| 4 + \frac{3}{2} + \frac{6}{4} - 6 \right| = 1$$

Коэффициент сбалансированности $K_b = 1$.

1.6. Информационная модель

Под информационной моделью понимается совокупность объектов ПО, их атрибутов и связей между ними. Она создается по принципу реляционной модели данных, т.е. представления данных в виде отношений между ними.

В информационной модели связи между объектами изображаются стрелками, указывающими направление связи. Возле рамки объекта, принимающего участие в связи, на линии стрелки указывается роль, которую

этот объект поддерживает в данной связи. Также существуют специальные обозначения для типов связей между объектами:

- 1:1 – двунаправленная стрелка, имеющей по одному "наконечнику" с каждой стороны;
- 1:N – двунаправленная стрелка, имеющая два "наконечника" со стороны объекта, который состоит в связи с несколькими объектами;
- N:M – двунаправленная стрелка, имеющая по два "наконечника" с каждой стороны.

Пример информационной модели с отображением связей для реализуемого ПО представлен на рис.6:

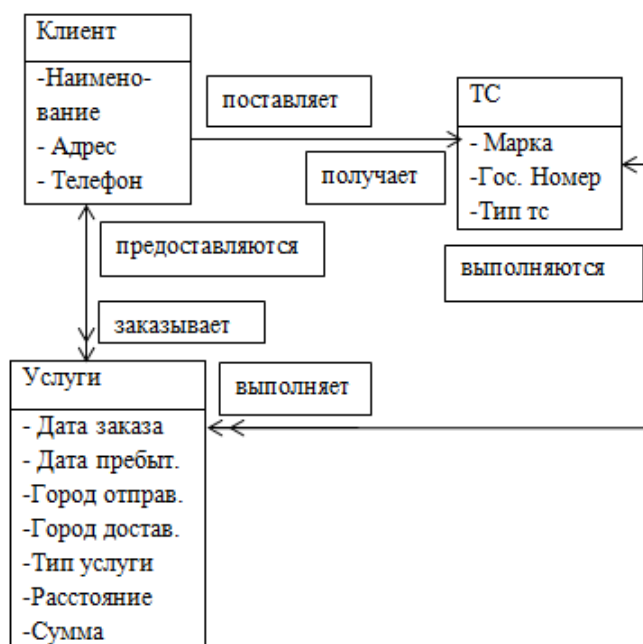


Рис.6. Информационная модель

1.7 Модель состояний

Модель состояний предназначена для отображения динамического поведения и изменения состояний поведения и изменения состояний каждого из объектов информационной модели и жизненного цикла поведения объектов. Состояние в модели – это положение или ситуация объекта, определяемая правилами и линией поведения.

Событие заставляет объект переходить из одного состояния в другое.
Экземпляры класса имеют поведение, которое определяется:

- состоянием, зависящим от текущих значений отдельных его атрибутов
- состоянием, изменяемым в результате выполненных над объектами действий
- состоянием ПО, зависящим от совокупности состояний ее объектов
- некоторыми процессами и действиями , которые изменяют жизненный цикл состояния объекта

Построение модели состояний начинается после выделения информационной модели отдельных объектов, обладающих динамическим поведением, создания экземпляра объекта или его уничтожения после прекращения существования.

Таблица 3. Модель переходов состояния

Услуга	Получение заказа	Выполнение	Получение усл.
Делаем заказ	2	Игнорируем	Ожидание
Выполняем	Игнорируем	3	Ожидание
Получаем усл.	Игнорируем	Игнорируем	1

1.8. Методы проектирования архитектуры ПО

Архитектура системы – это структурная схема компонентов системы, взаимодействующих между собой через интерфейсы. Основным условием построения архитектуры системы является декомпозиция системы на компоненты или модули. Кроме того, существует необходимость определения целей и проверка их выполнимости, определение входных и выходных данных и иерархическое представление абстракции системы.

Фактически создаваемая архитектура состоит из четырех уровней:

- 1-й уровень – системные компоненты.
- 2-й уровень – общесистемные компоненты.
- 3-й уровень – специфические компоненты определенной проблемной области.
- 4-й уровень – прикладные программные системы, реализуют конкретные задачи отдельных групп потребителей.

При проектировании архитектуры программная система рассматривается как композиция компонент третьего уровня.

Результатом архитектурного проектирования представляются нотациями в виде *диаграмм* (сущность-связь, переходы состояний, потоки данных и действий и т.п.).

Структурные (structural) модели:

- Диаграммы классов (class diagrams)
- Диаграммы компонентов (component diagrams)
- Диаграммы размещения (deployment diagrams)

Модели поведения (behavioral):

- Диаграммы вариантов использования (use case diagrams)
- Диаграммы взаимодействия (interaction diagrams)
- Диаграммы последовательности (sequence diagrams)
- Диаграммы состояний (statechart diagrams)
- Диаграммы деятельности (activity diagrams)

Рассмотрим их подробнее.

1.8.1. Диаграмма классов

Данная диаграмма используется для моделирования статической структуры классов системы и связей между ними. На рис.7 изображен пример диаграммы классов для разрабатываемого ПО.

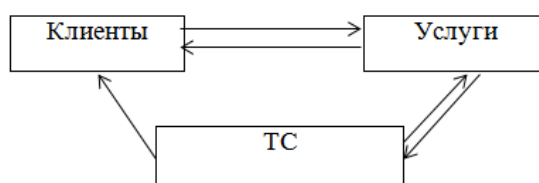


Рис.7. Диаграмма классов

Классы: услуги, ТС, клиент

1.8.2. Диаграмма вариантов использования

Эта диаграмма используется для моделирования процессов и функциональных требований к создаваемой системе. На рис. 8 представлена диаграмма вариантов использования для разрабатываемого ПО.

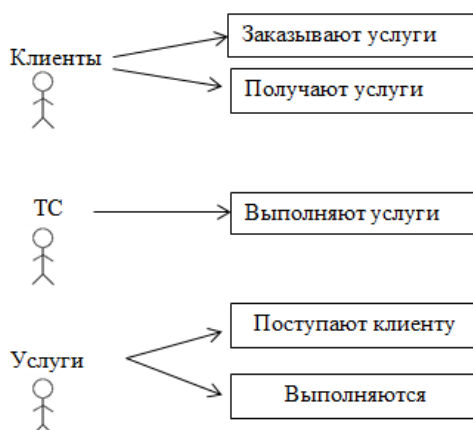


Рис.8. Диаграмма вариантов использования

1.8.3. Диаграмма последовательности

Диаграмма последовательности применяется для задания взаимодействия объектов, с помощью сценариев, отображающих события, связанные с их созданием и уничтожением. Взаимодействие объектов контролируется событиями, которые происходят в сценарии и поддерживаются сообщениями к другим объектам (рис 9).

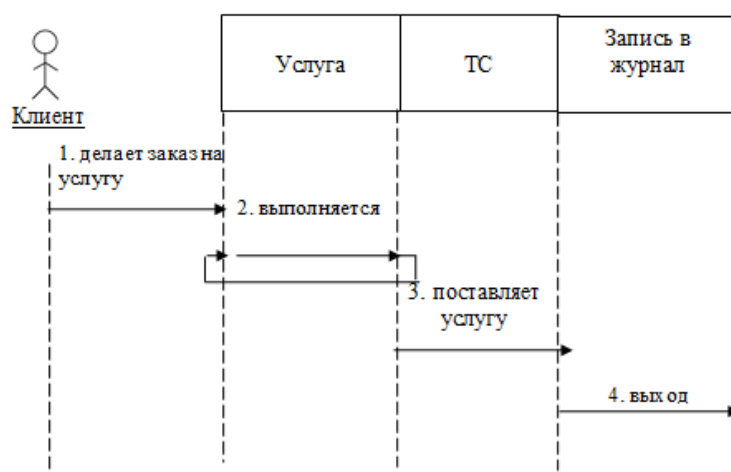


Рис.9. Диаграмма последовательности

1.8.4. Кооперативная диаграмма

На кооперативных диаграммах объекты (или классы) показываются в виде прямоугольников, а стрелками обозначаются сообщения, которыми они обмениваются в рамках одного варианта использования. Временная последовательность сообщений отражается их нумерацией (рис.10).

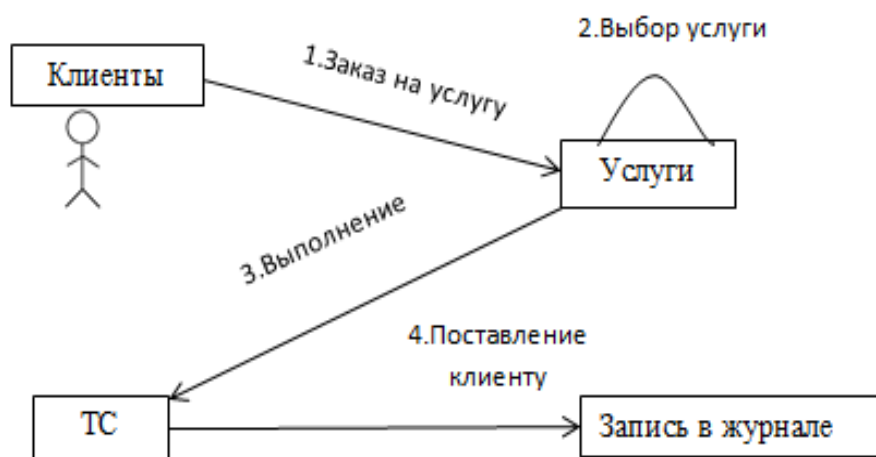


Рис.10. Кооперативная диаграмма

1.8.5. Диаграмма деятельности

Диаграмма деятельности применяется для моделирования поведения системы в рамках различных вариантов использования, или потоков управления (рис.11).

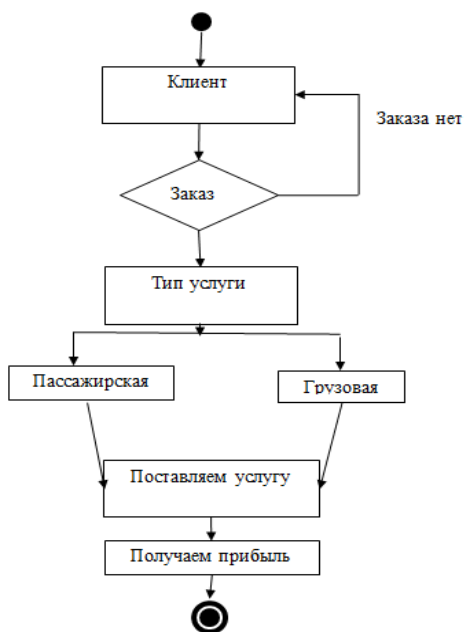


Рис.11. Диаграмма деятельности

2. ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

2.1. Инструкция пользователю

Реализация программы и разработка приложения осуществлялась в программе «*Microsoft Visual Studio C# 2010*». Основой алгоритма приняты разработанные выше схемы. Итоговый вид формы в режиме конструктора представлен на рис. 12-14.

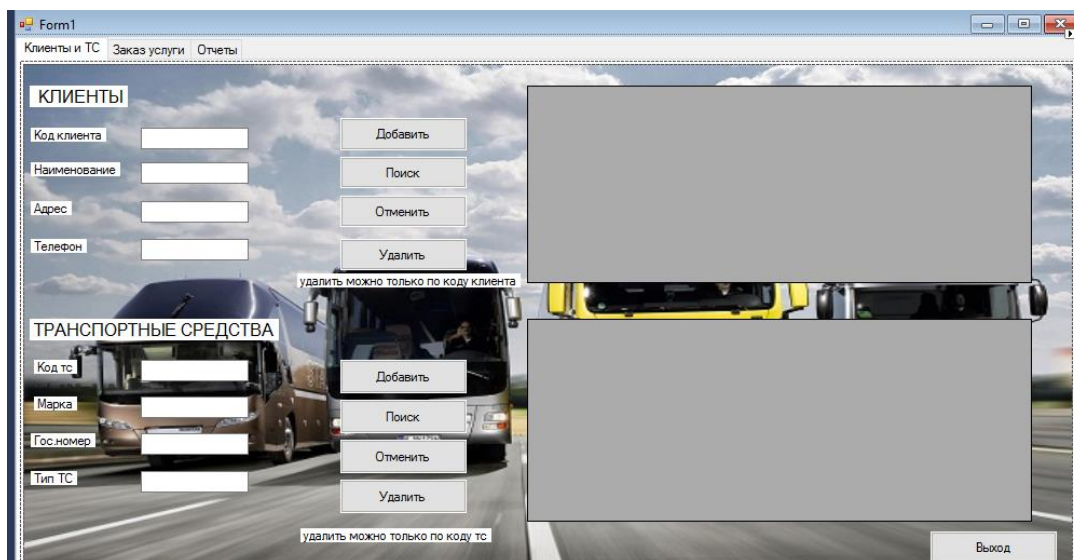


Рис.12. Закладка «Клиенты ТС» в режиме конструктора

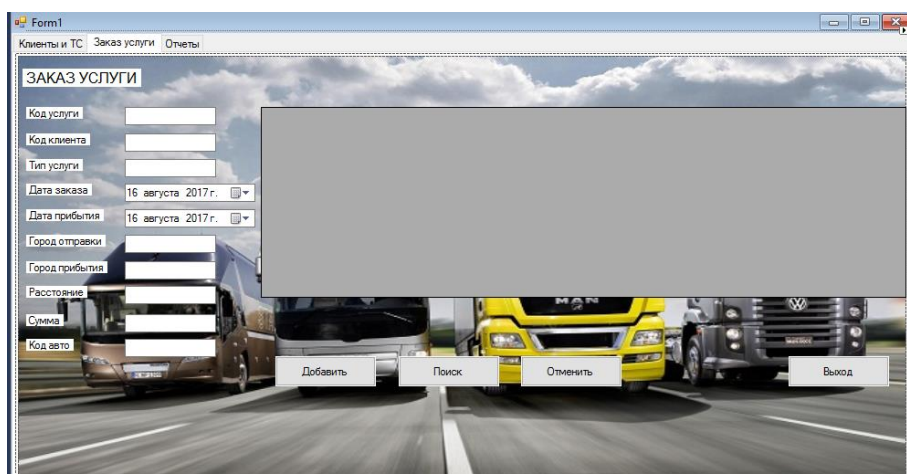


Рис.13. Закладка «Заказ услуги» в режиме конструктора



Рис.14. Закладка «Отчеты» в режиме конструктора

После запуска приложения, пользователь имеет право выбрать информацию, которую он хочет увидеть на экране с помощью соответствующих закладок.

1.Закладка «Клиенты и ТС»

Вид приложения после запуска представлено на рис.15.

The screenshot shows a software application window titled 'Form1' with a tabbed interface. The active tab is 'Клиенты и ТС', with other tabs being 'Заказ услуги' and 'Отчеты'. The background of the window features a bus on a road.

КЛИЕНТЫ

Input fields: Код клиента, Наименование, Адрес, Телефон. Buttons: Добавить, Поиск, Отменить, Удалить. A tooltip states: 'удалить можно только по коду клиента'.

Код_клиента	Наименование	Адрес	Телефон
1	ООО ПензаИнв...	Мира 51	89656311432
2	Квант-ИТ	Пролетарская 5	89063156784
3	ДШИ Радуга	Рахманинова 13	89093216572
4	ООО Лилия	Кихеватова 87	89633761342
5	ДМШ №1	Кирова 4	89065654234
6	ЗАО ТКП	Луначарского 3	89374320000
7	ЗАО Авант...	Мира 51	89373156784

ТРАНСПОРТНЫЕ СРЕДСТВА

Input fields: Код ТС, Марка, Гос.номер, Тип ТС. Buttons: Добавить, Поиск, Отменить, Удалить. A tooltip states: 'удалить можно только по коду ТС'.

Код_ТС	Марка	Гос_номер	Тип_ТС
1	Mitsubishi Rosa	о310ак	автобус
2	Scania OmniCity	у019ур	автобус
3	Volvo FM9	н770нн	грузовик
4	ЛиАЗ 5293	а626рк	автобус
5	Mack VISION	р070вк	грузовик
6	MAZ 6501A5	а798ар	грузовик
7	Kia Kosmos	м976мм	автобус

Выход

Рис.15. Закладка «Клиенты и ТС»

С выведенными таблицами можно выполнить несколько действий.

1) Добавление данных в таблицу «Клиенты». Для этого необходимо ввести данные клиента и нажать кнопку «Добавить».

КЛИЕНТЫ

Код клиента: 8
 Наименование: ООО Зима
 Адрес: Калинина 10
 Телефон: 89075432687

Добавить
Поиск
Отменить
Удалить

удалить можно только по коду клиента

ТРАНСПОРТНЫЕ СРЕДСТВА

Код ТС:
 Марка:
 Гос.номер:
 Тип ТС:

Добавить
Поиск
Отменить
Удалить

удалить можно только по коду ТС

Код_клиента	Наименование	Адрес	Телефон
2	Квант-ИТ	Пролетарская 5	89063156784
3	ДШИ Радуга	Рахманинова 13	89093216572
4	ООО Лилия	Кичеватова 87	89633761342
5	ДМШ №1	Кирова 4	89065654234
6	ЗАО ТКП	Луначарского 3	89374320000
7	ЗАО Аком	Маркина 8	89273456793

Код_ТС	Марка	Гос_номер	Тип_ТС
1	Mitsubishi Rosa	о310ак	автобус
2	Scania OmniCity	у019ур	автобус
3	Volvo FM9	н770нн	грузовик
4	ЛиАЗ 5293	а626рк	автобус
5	Mack VISION	р070вк	грузовик
6	МАЗ 6501А5	а798ар	грузовик
7	Kia Kosmos	м976мм	автобус

Выход

Рис.16. Вводим данные

Пример результата добавления представлен на рис.17.

КЛИЕНТЫ

Код клиента:
 Наименование:
 Адрес:
 Телефон:

Добавить
Поиск
Отменить
Удалить

удалить можно только по коду клиента

ТРАНСПОРТНЫЕ СРЕДСТВА

Код ТС:
 Марка:
 Гос.номер:
 Тип ТС:

Добавить
Поиск
Отменить
Удалить

удалить можно только по коду ТС

Код_клиента	Наименование	Адрес	Телефон
3	ДШИ Радуга	Рахманинова 13	89093216572
4	ООО Лилия	Кичеватова 87	89633761342
5	ДМШ №1	Кирова 4	89065654234
6	ЗАО ТКП	Луначарского 3	89374320000
7	ЗАО Аком	Маркина 8	89273456793
8	ООО Зима	Калинина 10	89075432687

Код_ТС	Марка	Гос_номер	Тип_ТС
1	Mitsubishi Rosa	о310ак	автобус
2	Scania OmniCity	у019ур	автобус
3	Volvo FM9	н770нн	грузовик
4	ЛиАЗ 5293	а626рк	автобус
5	Mack VISION	р070вк	грузовик
6	МАЗ 6501А5	а798ар	грузовик
7	Kia Kosmos	м976мм	автобус

Выход

Рис.17. Результаты добавления

Добавление данных в таблицу «ТС». Для этого необходимо ввести данные ТС и нажать кнопку «Добавить».

КЛИЕНТЫ

Код клиента: Добавить

Наименование: Поиск

Адрес: Отменить

Телефон: Удалить

удалить можно только по коду клиента

ТРАНСПОРТНЫЕ СРЕДСТВА

Код ТС: Добавить

Марка: Поиск

Гос.номер: Отменить

Тип ТС: Удалить

удалить можно только по коду ТС

Код_клиента	Наименование	Адрес	Телефон
3	ДШИ Радуга	Рахманинова 13	89093216572
4	ООО Лилия	Кижеватова 87	89633761342
5	ДМШ №1	Кирова 4	89065654234
6	ЗАО ТКП	Луначарского 3	89374320000
7	ЗАО Аком	Маркина 8	89273456793
8	ООО Зима	Калинина 10	89075432687

Код_ТС	Марка	Гос_номер	Тип_ТС
1	Mitsubishi Rosa	о310ак	автобус
2	Scania OmniCity	у019ур	автобус
3	Volvo FM9	н770нн	грузовик
4	ЛиАЗ 5293	а626рк	автобус
5	Mack VISION	р070вк	грузовик
6	MAZ 6501A5	а798ар	грузовик
7	Kia Kosmos	м976мм	автобус

Выход

Рис.18. Добавление нового ТС

КЛИЕНТЫ

Код клиента: Добавить

Наименование: Поиск

Адрес: Отменить

Телефон: Удалить

удалить можно только по коду клиента

ТРАНСПОРТНЫЕ СРЕДСТВА

Код ТС: Добавить

Марка: Поиск

Гос.номер: Отменить

Тип ТС: Удалить

удалить можно только по коду ТС

Код_клиента	Наименование	Адрес	Телефон
3	ДШИ Радуга	Рахманинова 13	89093216572
4	ООО Лилия	Кижеватова 87	89633761342
5	ДМШ №1	Кирова 4	89065654234
6	ЗАО ТКП	Луначарского 3	89374320000
7	ЗАО Аком	Маркина 8	89273456793
8	ООО Зима	Калинина 10	89075432687

Код_ТС	Марка	Гос_номер	Тип_ТС
4	ЛиАЗ 5293	а626рк	автобус
5	Mack VISION	р070вк	грузовик
6	MAZ 6501A5	а798ар	грузовик
7	Kia Kosmos	м976мм	автобус
8	Ford 2530D	у364ум	грузовик
9	Voivo FM9	р967кт	грузовик

Выход

Рис.19. Результаты добавления

2) **Поиск по таблице «Клиенты».** Для этого необходимо ввести данные в любое поле таблицы. После ввода данных необходимо нажать на кнопку «Поиск». Пример результата поиска представлен на рис. 20.

КЛИЕНТЫ

Код клиента: 5

Наименование:

Адрес:

Телефон:

Добавить

Поиск

Отменить

Удалить

удалить можно только по коду клиента

Код_клиента	Наименование	Адрес	Телефон	Edit
5	ДМШ №1	Кирова 4	89065654234	

ТРАНСПОРТНЫЕ СРЕДСТВА

Код ТС:

Марка:

Гос.номер:

Тип ТС:

Добавить

Поиск

Отменить

Удалить

удалить можно только по коду ТС

Код_ТС	Марка	Гос.номер	Тип_ТС
4	ЛиАЗ 5293	а626рк	автобус
5	Mack VISION	р070вк	грузовик
6	МАЗ 6501А5	а798ар	грузовик
7	Киа Kozmos	м976мм	автобус
8	Ford 2530D	у364ум	грузовик
9	Voivo FM9	р967кт	грузовик

Выход

Рис.20. Результаты поиска

После можно нажать кнопку «Отменить» и ввести новый параметр поиска (рис.21).

КЛИЕНТЫ

Код клиента:

Наименование: ДШИ Радуга

Адрес:

Телефон:

Добавить

Поиск

Отменить

Удалить

удалить можно только по коду клиента

Код_клиента	Наименование	Адрес	Телефон	Edit
3	ДШИ Радуга	Рахманинова 13	89093216572	

ТРАНСПОРТНЫЕ СРЕДСТВА

Код ТС:

Марка:

Гос.номер:

Тип ТС:

Добавить

Поиск

Отменить

Удалить

удалить можно только по коду ТС

Код_ТС	Марка	Гос.номер	Тип_ТС
4	ЛиАЗ 5293	а626рк	автобус
5	Mack VISION	р070вк	грузовик
6	МАЗ 6501А5	а798ар	грузовик
7	Киа Kozmos	м976мм	автобус
8	Ford 2530D	у364ум	грузовик
9	Voivo FM9	р967кт	грузовик

Выход

Рис.21. Результаты поиска

Поиск по таблице «ТС». Для этого необходимо ввести данные в любое поле таблицы. После ввода данных необходимо нажать на кнопку «Поиск» (рис.22).

КЛИЕНТЫ

Код клиента:

Наименование:

Адрес:

Телефон:

удалить можно только по коду клиента

ТРАНСПОРТНЫЕ СРЕДСТВА

Код ТС:

Марка:

Гос.номер:

Тип ТС:

удалить можно только по коду ТС

Код_клиента	Наименование	Адрес	Телефон
1	ООО ПензаИнв...	Мира 51	89656311432
2	Квант-ИТ	Пролетарская 5	89063156784
3	ДШИ Радуга	Рахманинова 13	89093216572
4	ООО Лилия	Кихеватова 87	89633761342
5	ДМШ №1	Кирова 4	89065654234
6	ЗАО ТКП	Луначарского 3	89374320000
7	ЗАО Акком	Маркина 8	89273456793

Код_ТС	Марка	Гос_номер	Тип_ТС
4	ЛиАЗ 5293	а626рк	автобус

Выход

Рис.22. Результаты поиска

После можно нажать кнопку «Отменить» и ввести новый параметр поиска.

3) **Удаление в таблице «Клиенты».** Для этого необходимо ввести данные в поле «Код клиента» (рис.23).

КЛИЕНТЫ

Код клиента:

Наименование:

Адрес:

Телефон:

удалить можно только по коду клиента

ТРАНСПОРТНЫЕ СРЕДСТВА

Код ТС:

Марка:

Гос.номер:

Тип ТС:

удалить можно только по коду ТС

Код_клиента	Наименование	Адрес	Телефон
3	ДШИ Радуга	Рахманинова 13	89093216572
4	ООО Лилия	Кихеватова 87	89633761342
5	ДМШ №1	Кирова 4	89065654234
6	ЗАО ТКП	Луначарского 3	89374320000
7	ЗАО Акком	Маркина 8	89273456793
8	ООО Зима	Калинина 10	89075432687

Код_ТС	Марка	Гос_номер	Тип_ТС
1	Mitsubishi Rosa	о310ак	автобус
2	Scania OmniCity	у019ур	автобус
3	Volvo FM9	н770нн	грузовик
4	ЛиАЗ 5293	а626рк	автобус
5	Mack VISION	р070вк	грузовик
6	МАЗ 6501А5	а798ар	грузовик
7	Kia K200S	м976мм	автобус

Выход

Рис.23. Вводим данные

После ввода данных необходимо нажать на кнопку «Удалить». Пример результата удаления представлен на рис. 24.

КЛИЕНТЫ

Код клиента: Добавить

Наименование: Поиск

Адрес: Отменить

Телефон: **Удалить**

удалить можно только по коду клиента

ТРАНСПОРТНЫЕ СРЕДСТВА

Код ТС: Добавить

Марка: Поиск

Гос.номер: Отменить

Тип ТС: Удалить

удалить можно только по коду ТС

Код_клиента	Наименование	Адрес	Телефон
2	Квант-ИТ	Пролетарская 5	89063156784
3	ДШИ Радуга	Рахманинова 13	89093216572
4	ООО Лилия	Кихеятова 87	89633761342
5	ДМШ №1	Кирова 4	89065654234
6	ЗАО ТКП	Луначарского 3	89374320000
7	ЗАО Аком	Маркина 8	89273456793

Код_ТС	Марка	Гос_номер	Тип_ТС
1	Mitsubishi Rosa	о310ак	автобус
2	Scania OmniCity	у019ур	автобус
3	Volvo FM9	н770нн	грузовик
4	ЛиАЗ 5293	а626рк	автобус
5	Mack VISION	р070вк	грузовик
6	MAZ 6501A5	а798ар	грузовик
7	Kia Kosmos	м976мм	автобус

Выход

Рис.24. Результат удаления

Удаление в таблице «ТС». Для этого необходимо ввести данные в поле «Код ТС» (рис.25).

КЛИЕНТЫ

Код клиента: Добавить

Наименование: Поиск

Адрес: Отменить

Телефон: Удалить

удалить можно только по коду клиента

ТРАНСПОРТНЫЕ СРЕДСТВА

Код ТС: Добавить

Марка: Поиск

Гос.номер: Отменить

Тип ТС: Удалить

удалить можно только по коду ТС

Код_клиента	Наименование	Адрес	Телефон
2	Квант-ИТ	Пролетарская 5	89063156784
3	ДШИ Радуга	Рахманинова 13	89093216572
4	ООО Лилия	Кихеятова 87	89633761342
5	ДМШ №1	Кирова 4	89065654234
6	ЗАО ТКП	Луначарского 3	89374320000
7	ЗАО Аком	Маркина 8	89273456793

Код_ТС	Марка	Гос_номер	Тип_ТС
1	Mitsubishi Rosa	о310ак	автобус
2	Scania OmniCity	у019ур	автобус
3	Volvo FM9	н770нн	грузовик
4	ЛиАЗ 5293	а626рк	автобус
5	Mack VISION	р070вк	грузовик
6	MAZ 6501A5	а798ар	грузовик
7	Kia Kosmos	м976мм	автобус

Выход

Рис.25. Вводим данные

После ввода данных необходимо нажать на кнопку «Удалить». Пример результата удаления представлен на рис. 26.

КЛИЕНТЫ

Код клиента: Добавить

Наименование: Поиск

Адрес: Отменить

Телефон: Удалить

удалить можно только по коду клиента

Код_клиента	Наименование	Адрес	Телефон
2	Квант-ИТ	Пролетарская 5	89063156784
3	ДШИ Радуга	Рахманинова 13	89093216572
4	ООО Пилия	Кижеватова 87	89633761342
5	ДМШ №1	Кирова 4	89065654234
6	ЗАО ТКП	Луначарского 3	89374320000
7	ЗАО Аком	Маркина 8	89273456793

ТРАНСПОРТНЫЕ СРЕДСТВА

Код ТС: Добавить

Марка: Поиск

Гос.номер: Отменить

Тип ТС: Удалить

удалить можно только по коду ТС

Код_ТС	Марка	Гос.номер	Тип_ТС
3	Volvo FM9	н770нн	грузовик
4	ЛиАЗ 5293	а626рк	автобус
5	Mack VISION	р070вк	грузовик
6	МАЗ 6501А5	а798ар	грузовик
7	Kia Kosmos	м976мм	автобус
8	Ford 2530D	у364ум	грузовик

Выход

Рис.26. Результат удаления

2. Закладка «Заказ услуги»

Вид приложения после запуска представлено на рис.27.

ЗАКАЗ УСЛУГИ

Код услуги:

Код клиента:

Тип услуги:

Дата заказа: 16 августа 2017 г.

Дата прибытия: 16 августа 2017 г.

Город отправки:

Город прибытия:

Расстояние:

Сумма:

Код авто:

Код_услуги	Код_клиента	Тип_услуги	Дата_заказа	Дата_прибытия	Город_отправки	Город_прибытия
1	1	груз	04.07.2017	11.07.2017	Тула	Омск
2	2	груз	06.07.2017	13.07.2017	Самара	Тамбов
3	3	пассажир	10.07.2017	14.07.2017	Москва	Пенза
4	4	груз	10.07.2017	18.07.2017	Липецк	Краснодар
5	5	пассажир	11.07.2017	16.07.2017	Смоленск	Курск
6	1	груз	13.07.2017	20.07.2017	Магнитогорск	Пенза

Добавить Поиск Отменить Выход

Рис.27. Закладка «Заказ услуги»

С выведенной таблицей можно выполнить несколько действий.

1) Добавление данных в таблицу «Услуги». Для этого необходимо ввести данные услуги и нажать кнопку «Добавить».

The screenshot shows the 'Form1' application window with the 'Заказ услуги' tab selected. The form contains input fields for service details, and a table displays existing services. The background image shows a highway with several trucks.

Код услуги	Код клиента	Тип услуги	Дата заказа	Дата прибытия	Город отправки	Город прибытия
1	1	груз	04.07.2017	11.07.2017	Тула	Омск
2	2	груз	06.07.2017	13.07.2017	Самара	Тамбов
3	3	пассажир	10.07.2017	14.07.2017	Москва	Пенза
4	4	груз	10.07.2017	18.07.2017	Липецк	Краснодар
5	5	пассажир	11.07.2017	16.07.2017	Смоленск	Курск
6	1	груз	13.07.2017	20.07.2017	Магнитогорск	Пенза

Рис.28. Вводим данные

Пример результата добавления представлен на рис.29.

The screenshot shows the 'Form1' application window after adding a new service. The table now contains 7 entries. The background image remains the same highway scene with trucks.

Код услуги	Код клиента	Тип услуги	Дата заказа	Дата прибытия	Город отправки	Город прибытия
1	1	груз	04.07.2017	11.07.2017	Тула	Омск
2	2	груз	06.07.2017	13.07.2017	Самара	Тамбов
3	3	пассажир	10.07.2017	14.07.2017	Москва	Пенза
4	4	груз	10.07.2017	18.07.2017	Липецк	Краснодар
5	5	пассажир	11.07.2017	16.07.2017	Смоленск	Курск
6	1	груз	13.07.2017	20.07.2017	Магнитогорск	Пенза
7	7	пассажир	15.07.2017 19:00	19.07.2017 19:00	Пенза	Москва

Рис.29. Результат добавления

2) **Поиск по таблице «Услуги».** Для этого необходимо ввести данные в любое поле таблицы. После ввода данных необходимо нажать на кнопку «Поиск». Пример результата поиска представлен на рис. 30.

The screenshot shows a software application window titled "Form1" with a menu bar containing "Клиенты и ТС", "Заказ услуги", and "Отчеты". The main area is titled "ЗАКАЗ УСЛУГИ" and contains several input fields on the left: "Код услуги" (value: 3), "Код клиента", "Тип услуги", "Дата заказа" (15 июля 2017 г.), "Дата прибытия" (19 июля 2017 г.), "Город отправки", "Город прибытия", "Расстояние", "Сумма", and "Код авто". On the right, a table displays search results. The first row is highlighted in blue and contains the following data:

Код услуги	Код клиента	Тип услуги	Дата заказа	Дата прибытия	Город отправки	Город прибытия
3	3	пассажир	10.07.2017	14.07.2017	Москва	Пенза

Below the table, there are four buttons: "Добавить", "Поиск", "Отменить", and "Выход". The background of the application window features a collage of various trucks and buses.

Рис.30. Результат поиска

После можно нажать кнопку «Отменить» и ввести новый параметр поиска (рис.31).

ЗАКАЗ УСЛУГИ

Код услуги:

Код клиента:

Тип услуги:

Дата заказа: 15 июля 2017 г.

Дата прибытия: 19 июля 2017 г.

Город отправки:

Город прибытия:

Расстояние:

Сумма:

Код авто:

Код_услуги	Код_клиента	Тип_услуги	Дата_заказа	Дата_прибытия	Город_отправки	Город_прибытия
2	2	груз	06.07.2017	13.07.2017	Самара	Тамбов
4	4	груз	10.07.2017	18.07.2017	Липецк	Краснодар
6	1	груз	13.07.2017	20.07.2017	Магнитогорск	Пенза
1	1	груз	04.07.2017	11.07.2017	Тула	Омск

Добавить Поиск Отменить Выход

Рис.31. Результат поиска

3. Закладка «Отчеты»

Вид приложения после запуска представлено на рис.32.

Клиенты и ТС Заказ услуги **Отчеты**

Тип

Сумма

Отчет 1 Отчет 2

Рис.32. Закладка «Отчеты»

1) Отчёт 1. При нажатии на кнопку **«Отчёт 1»** перед пользователем предстанет следующее окно (рис. 33). В нём будет отражена таблица

статистических данных типов услуг. Для наглядности вся информация представлена на диаграмме.

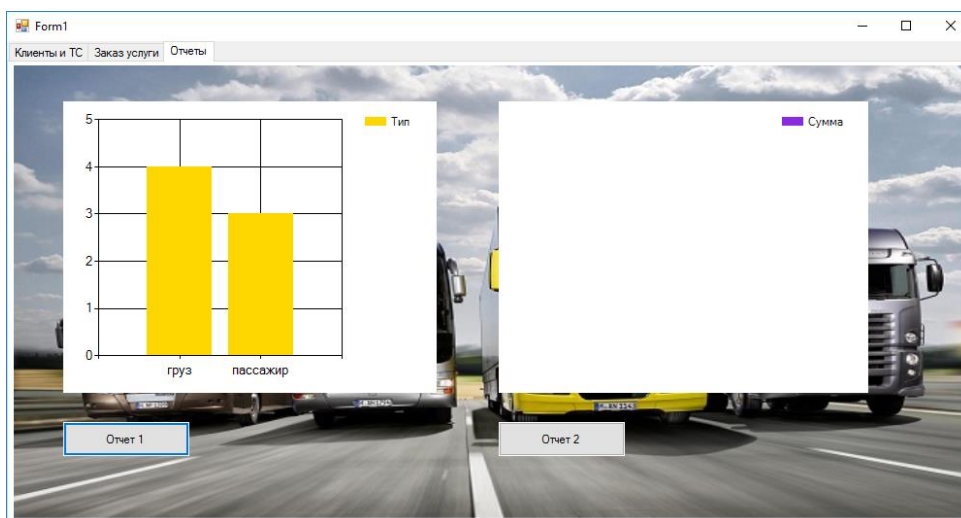


Рис.33. Вывод «Отчет 1»

2)Отчёт 2. При нажатии на кнопку «**Отчёт 2**» перед пользователем предстанет другое окно (рис.34). В нём будет отражена таблица статистических данных заказов услуг. На диаграмме отображается на какую сумму сделаны услуги.



Рис.34. Вывод « Отчет 2»

Чтобы посмотреть изменения графиков, добавим новый заказ на услугу (рис.35).

Form1

Клиенты и ТС Заказ услуги Отчеты

ЗАКАЗ УСЛУГИ

Код услуги: 8
Код клиента: 2
Тип услуги: груз
Дата заказа: 17 июля 2017 г.
Дата прибытия: 28 июня 2017 г.
Город отправки: Тамбов
Город прибытия: Самара
Расстояние: 705
Сумма: 42300
Код авто: 5

id_клиента	Тип_услуги	Дата_заказа	Дата_прибытия	Город_отправки	Город_прибытия	Расстояние
	груз	04.07.2017	11.07.2017	Тула	Омск	2780
	груз	06.07.2017	13.07.2017	Самара	Тамбов	705
	пассажир	10.07.2017	14.07.2017	Москва	Пенза	640
	груз	10.07.2017	18.07.2017	Липецк	Краснодар	952
	пассажир	11.07.2017	16.07.2017	Смоленск	Курск	534
	груз	13.07.2017	20.07.2017	Магнитогорск	Пенза	1450
	пассажир	15.07.2017 19:00	19.07.2017 19:00	Пенза	Москва	640

Добавить Поиск Отменить Выход

Рис.35. Добавление новой услуги

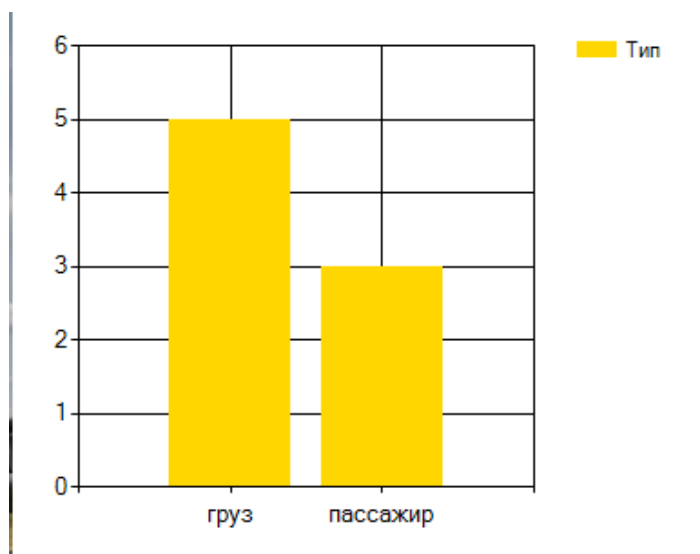


Рис.36. Результат изменения «Отчета 1»

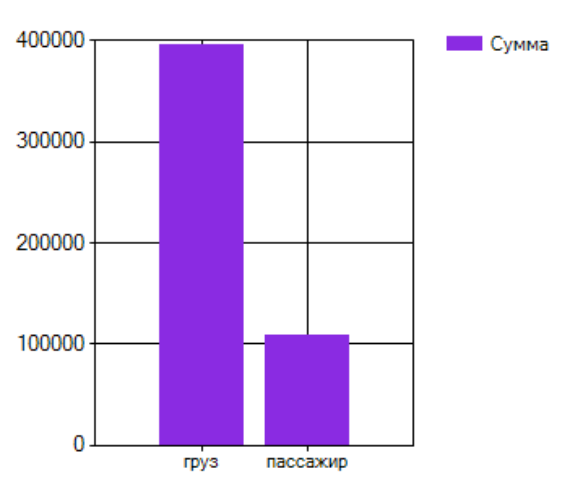


Рис.37. Результат изменения «Отчета 2»

ЗАКЛЮЧЕНИЕ

В ходе выполнения данной курсовой работы было реализовано приложение с помощью программы «*Microsoft Visual Studio C# 2010*». Целью которой была автоматизация системы учета услуг автотранспортного предприятия. Для зрительной реализации полученных результатов в программе использованы диаграммы.

Приложение

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.OleDb;

namespace АВТО
{
    public partial class Form1 : Form
    {
        public OleDbConnection database;
        DataGridViewButtonColumn editButton;
        DataGridViewButtonColumn deleteButton;

        public Form1()
        {
            InitializeComponent();
            string connectionString = "Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=предпр.accdb";

            try
            {
                database = new OleDbConnection(connectionString);
                database.Open();
                string queryString = " SELECT Код_клиента, Наименование, Адрес, Телефон
FROM Клиенты";
                loadDataGridView1(queryString);
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
                return;
            }

            try
            {
                database = new OleDbConnection(connectionString);
                database.Open();
                string queryString = " SELECT Код_тс, Марка, Гос_номер, Тип_тс FROM TC";
                loadDataGridView2(queryString);
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
                return;
            }

            try
            {
                database = new OleDbConnection(connectionString);
                database.Open();
                string queryString = " SELECT Код_услуги, Код_клиента,Тип_услуги,
Дата_заказа, Дата_прибытия, Город_отправки, Город_прибытия, Расстояние, Сумма, Код_тс
FROM Услуги";
```

```

        loadDataGridView3(queryString);
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
        return;
    }
}

private void dataGridView1_CellContentClick(object sender,
DataGridViewCellEventArgs e)
{
}

public void loadDataGridView1(string sqlQueryString)//отображает клиентов
{
    OleDbCommand SQLQuery = new OleDbCommand();
    DataTable data = null;
    dataGridView1.DataSource = null;
    SQLQuery.Connection = null;
    OleDbDataAdapter dataAdapter = null;
    dataGridView1.Columns.Clear();
    //-----
    SQLQuery.CommandText = sqlQueryString;
    SQLQuery.Connection = database;
    data = new DataTable();
    dataAdapter = new OleDbDataAdapter(SQLQuery);
    dataAdapter.Fill(data);
    dataGridView1.DataSource = data;
    dataGridView1.AllowUserToAddRows = false;
    dataGridView1.ReadOnly = true;
    editButton = new DataGridViewButtonColumn();
    editButton.HeaderText = "Edit";
    editButton.Text = "Edit";
    editButton.UseColumnTextForButtonValue = true;
    editButton.Width = 80;
    dataGridView1.Columns.Add(editButton);
    deleteButton = new DataGridViewButtonColumn();
}

public void loadDataGridView2(string sqlQueryString)//отображает TC
{
    OleDbCommand SQLQuery = new OleDbCommand();
    DataTable data = null;
    dataGridView2.DataSource = null;
    SQLQuery.Connection = null;
    OleDbDataAdapter dataAdapter = null;
    dataGridView2.Columns.Clear();
    //-----
    SQLQuery.CommandText = sqlQueryString;
    SQLQuery.Connection = database;
    data = new DataTable();
    dataAdapter = new OleDbDataAdapter(SQLQuery);
    dataAdapter.Fill(data);
    dataGridView2.DataSource = data;
    dataGridView2.AllowUserToAddRows = false;
    dataGridView2.ReadOnly = true;

    editButton = new DataGridViewButtonColumn();
    editButton.HeaderText = "Edit";

```

```

        editButton.Text = "Edit";
        editButton.UseColumnTextForButtonValue = true;
        editButton.Width = 80;
        dataGridView2.Columns.Add(editButton);
        deleteButton = new DataGridViewButtonColumn();
    }

    public void loadDataGridView3(string sqlQueryString)//отображает услуги
    {
        OleDbCommand SQLQuery = new OleDbCommand();
        DataTable data = null;
        dataGridView3.DataSource = null;
        SQLQuery.Connection = null;
        OleDbDataAdapter dataAdapter = null;
        dataGridView3.Columns.Clear();
        //-----
        SQLQuery.CommandText = sqlQueryString;
        SQLQuery.Connection = database;
        data = new DataTable();
        dataAdapter = new OleDbDataAdapter(SQLQuery);
        dataAdapter.Fill(data);
        dataGridView3.DataSource = data;
        dataGridView3.AllowUserToAddRows = false;
        dataGridView3.ReadOnly = true;

        editButton = new DataGridViewButtonColumn();
        editButton.HeaderText = "Edit";
        editButton.Text = "Edit";
        editButton.UseColumnTextForButtonValue = true;
        editButton.Width = 80;
        dataGridView3.Columns.Add(editButton);
        deleteButton = new DataGridViewButtonColumn();
    }

    private void button1_Click(object sender, EventArgs e)// Кнопка добавить клиента
    {
        string SQLString = "";
        string kod = textBox1.Text.ToString();
        string nazvanie = textBox2.Text.ToString();
        string adres = textBox3.Text.ToString();
        string telephon = textBox4.Text.ToString();

        if (kod != "")
        {
            if (nazvanie != "")
            {
                if (adres != "")
                {
                    if (telephon != "")
                    {
                        SQLString = "INSERT INTO Клиенты( Код_клиента, Наименование, Адрес, Телефон) VALUES('" + kod + "', '" + nazvanie + "', '" + adres + "', '" + telephon + "')";
                        OleDbCommand SQLCommand = new OleDbCommand();
                        SQLCommand.CommandText = SQLString;
                        SQLCommand.Connection = database;
                        int response = -1;
                    }
                }
            }
        }
    }

```



```

        counter = counter + 1;
        mark = 3;
    }
    if (textBox4.Text != "")
    {
        counter = counter + 1;
        mark = 4;
    }

    if (counter == 1)
    {
        switch (mark)
        {
            case 1:
                string queryString = "SELECT Код_клиента, Наименование, Адрес,
Телефон FROM Клиенты WHERE Клиенты.Код_клиента LIKE '" + textBox1.Text + "%'";

                loadDataGridView1(queryString);
                break;
            case 2:
                string queryString1 = "SELECT Код_клиента, Наименование, Адрес,
Телефон FROM Клиенты WHERE Клиенты.Наименование LIKE '" + textBox2.Text + "%'";
                loadDataGridView1(queryString1);
                break;
            case 3:
                string queryString2 = "SELECT Код_клиента, Наименование, Адрес,
Телефон FROM Клиенты WHERE Клиенты.Адрес LIKE '" + textBox3.Text + "%'";
                loadDataGridView1(queryString2);
                break;
            case 4:
                string queryString3 = "SELECT Код_клиента, Наименование, Адрес,
Телефон FROM Клиенты WHERE Клиенты.Телефон LIKE '" + textBox4.Text + "%'";
                loadDataGridView1(queryString3);
                break;
        }
    }
    else if (counter == 0)
        MessageBox.Show("Заполните одно поле");
    else
        MessageBox.Show("Заполните ТОЛЬКО одно поле");
}

private void button3_Click(object sender, EventArgs e)// Кнопка отмены для
клиента
{
    string queryString1 = "SELECT Код_клиента, Наименование, Адрес, Телефон FROM
Клиенты";
    loadDataGridView1(queryString1);
}

private void button4_Click(object sender, EventArgs e)//Кнопка удаления для
клиента
{
    string queryString1 = "SELECT Код_клиента, Наименование, Адрес, Телефон FROM
Клиенты";
    string queryDeleteString = "DELETE FROM Клиенты WHERE Код_клиента = " +
textBox1.Text + "";
    OleDbCommand sqlDelete = new OleDbCommand();
    sqlDelete.CommandText = queryDeleteString;
    sqlDelete.Connection = database;
    sqlDelete.ExecuteNonQuery();
    loadDataGridView1(queryString1);
}

```

```

        dataGridView1.Update();
        textBox1.Text = "";
    }

private void button5_Click(object sender, EventArgs e) // Кнопка добавления для ТС
{
    string SQLString = "";
    string kod_ts = textBox5.Text.ToString();
    string marka = textBox6.Text.ToString();
    string gos = textBox7.Text.ToString();
    string tip = textBox8.Text.ToString();

    if (kod_ts != "")
    {
        if (marka != "")
        {
            if (gos != "")
            {
                if (tip != "")
                {
                    SQLString = "INSERT INTO TC( Код_тс, Марка, Гос_номер,
Тип_тс) VALUES('" + kod_ts + "', '" + marka + "', '" + gos + "', '" + tip + "')";
                    OleDbCommand SQLCommand = new OleDbCommand();
                    SQLCommand.CommandText = SQLString;
                    SQLCommand.Connection = database;
                    int response = -1;
                    try
                    {
                        response = SQLCommand.ExecuteNonQuery();
                    }
                    catch (Exception ex)
                    {
                        MessageBox.Show(ex.Message);
                    }
                    textBox5.Text = "";
                    textBox6.Text = "";
                    textBox7.Text = "";
                    textBox8.Text = "";

                    string queryString = "SELECT Код_тс, Марка, Гос_номер,
Тип_тс FROM TC";
                    loadDataGridView2(queryString);
                }
            }
        }
        else
        {
            MessageBox.Show("Заполните все поля");
            return;
        }
    }
    else
    {
        MessageBox.Show("Заполните все поля");
        return;
    }
}
else
{
    MessageBox.Show("Заполните все поля");
    return;
}
}

```

```

    }

}
else
{
    MessageBox.Show("Заполните все поля");
    return;
}

MessageBox.Show("ТС успешно добавлено");
}

private void button6_Click(object sender, EventArgs e)//Кнопка поиска ТС
{
    int counter = 0;
    int mark = 0;
    if (textBox5.Text != "")
    {
        counter = counter + 1;
        mark = 1;
    }
    if (textBox6.Text != "")
    {
        counter = counter + 1;
        mark = 2;
    }
    if (textBox7.Text != "")
    {
        counter = counter + 1;
        mark = 3;
    }
    if (textBox8.Text != "")
    {
        counter = counter + 1;
        mark = 4;
    }

    if (counter == 1)
    {
        switch (mark)
        {
            case 1:
                string queryString = "SELECT Код_тс, Марка, Гос_номер, Тип_тс
FROM TC WHERE TC.Код_тс LIKE '" + textBox5.Text + "%'";

                loadDataGridView2(queryString);
                break;
            case 2:
                string queryString1 = "SELECT Код_тс, Марка, Гос_номер, Тип_тс
FROM TC WHERE TC. Марка LIKE '" + textBox6.Text + "%'";
                loadDataGridView2(queryString1);
                break;
            case 3:
                string queryString2 = "SELECT Код_тс, Марка, Гос_номер, Тип_тс
FROM TC WHERE TC.Гос_номер LIKE '" + textBox7.Text + "%'";
                loadDataGridView2(queryString2);
                break;
            case 4:
                string queryString3 = "SELECT Код_тс, Марка, Гос_номер, Тип_тс
FROM TC WHERE TC.Тип_тс LIKE '" + textBox8.Text + "%'";
                loadDataGridView2(queryString3);
                break;
        }
    }
}

```

```

    }
}
else if (counter == 0)
    MessageBox.Show("Заполните одно поле");
else
    MessageBox.Show("Заполните ТОЛЬКО одно поле");
}

private void button7_Click(object sender, EventArgs e)//Кнопка отмены для ТС
{
    string queryString1 = "SELECT Код_тс, Марка, Гос_номер, Тип_тс FROM ТС";
    loadDataGridView2(queryString1);
}

private void button8_Click(object sender, EventArgs e)//Кнопка удалить для ТС
{
    string queryString1 = "SELECT Код_тс, Марка, Гос_номер, Тип_тс FROM ТС";
    string queryDeleteString = "DELETE FROM ТС WHERE Код_тс = " + textBox5.Text +
"";

    OleDbCommand sqlDelete = new OleDbCommand();
    sqlDelete.CommandText = queryDeleteString;
    sqlDelete.Connection = database;
    sqlDelete.ExecuteNonQuery();
    loadDataGridView2(queryString1);
    dataGridView2.Update();
    textBox5.Text = "";
}

private void button9_Click(object sender, EventArgs e)//Выход
{
    Application.Exit();
}

private void button10_Click(object sender, EventArgs e)//Кнопка добавить заказ
{
    string SQLString = "";
    string kod_usl = textBox9.Text.ToString();
    string kod_kl = textBox10.Text.ToString();
    string tip_usl = textBox11.Text.ToString();
    DateTime date_zakaza = dateTimePicker1.Value;
    DateTime date_okon = dateTimePicker2.Value;
    string gor_otp = textBox14.Text.ToString();
    string gor_dos = textBox15.Text.ToString();
    string rass = textBox16.Text.ToString();
    string sum = textBox17.Text.ToString();
    string kod_av = textBox18.Text.ToString();

    if (kod_usl != "")
    {
        if (kod_kl != "")
        {
            if (tip_usl != "")
            {
                if (gor_otp != "")
                {
                    if (gor_dos != "")
                    {
                        if (rass != "")
                        {
                            if (sum != "")

```

```

        {
            if (kod_av != "")
            {
                SQLString = "INSERT INTO Услуги( Код_услуги,
Код_клиента,Тип_услуги, Дата_заказа, Дата_прибытия, Город_отправки, Город_прибытия,
Расстояние, Сумма, Код_тс) VALUES('" + kod_usl + "', '" + kod_kl + "', '" + tip_usl + "',
'" + date_zakaza + "', '" + date_okon + "', '" + gor_otp + "', '" + gor_dos + "', '" +
rass + "', '" + sum + "', '" + kod_av + "')";
                OleDbCommand SQLCommand = new OleDbCommand();
                SQLCommand.CommandText = SQLString;
                SQLCommand.Connection = database;
                int response = -1;
                try
                {
                    response = SQLCommand.ExecuteNonQuery();
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message);
                }
                textBox9.Text = "";
                textBox10.Text = "";
                textBox11.Text = "";
                textBox14.Text = "";
                textBox15.Text = "";
                textBox16.Text = "";
                textBox17.Text = "";
                textBox18.Text = "";
                string queryString = "SELECT Код_услуги,
Код_клиента,Тип_услуги, Дата_заказа, Дата_прибытия, Город_отправки, Город_прибытия,
Расстояние, Сумма, Код_тс FROM Услуги";

                loadDataGridView3(queryString);
            }
            else
            {
                MessageBox.Show("Заполните все поля");
                return;
            }
        }
        else
        {
            MessageBox.Show("Заполните все поля");
            return;
        }
    }
    else
    {
        MessageBox.Show("Заполните все поля");
        return;
    }
}
else
{
    MessageBox.Show("Заполните все поля");
    return;
}

```

```

        }
    }
    else
    {
        MessageBox.Show("Заполните все поля");
        return;
    }
}
else
{
    MessageBox.Show("Заполните все поля");
    return;
}

}
else
{
    MessageBox.Show("Заполните все поля");
    return;
}

}
MessageBox.Show("Ваш заказ успешно добавлен");
}

private void button11_Click(object sender, EventArgs e)//Кнопка поиска услуг
{
    int counter = 0;
    int mark = 0;
    if (textBox9.Text != "")
    {
        counter = counter + 1;
        mark = 1;
    }
    if (textBox10.Text != "")
    {
        counter = counter + 1;
        mark = 2;
    }
    if (textBox11.Text != "")
    {
        counter = counter + 1;
        mark = 3;
    }

    if (textBox14.Text != "")
    {
        counter = counter + 1;
        mark = 6;
    }
    if (textBox15.Text != "")
    {
        counter = counter + 1;
        mark = 7;
    }
    if (textBox16.Text != "")
    {
        counter = counter + 1;
        mark = 8;
    }
    if (textBox17.Text != "")
    {
        counter = counter + 1;
        mark = 9;
    }
}

```

```

if (textBox18.Text != "")
{
    counter = counter + 1;
    mark = 10;
}

if (counter == 1)
{
    switch (mark)
    {
        case 1:
            string queryString = "SELECT Код_услуги, Код_клиента,Тип_услуги,
Дата_заказа, Дата_прибытия, Город_отправки, Город_прибытия, Расстояние, Сумма, Код_тс
FROM Услуги WHERE Услуги.Код_услуги LIKE '" + textBox9.Text + "%'";

            loadDataGridView3(queryString);
            break;
        case 2:
            string queryString1 = "SELECT Код_услуги,
Код_клиента,Тип_услуги, Дата_заказа, Дата_прибытия, Город_отправки, Город_прибытия,
Расстояние, Сумма, Код_тс FROM Услуги WHERE Услуги.Код_клиента LIKE '" + textBox10.Text +
"%'";

            loadDataGridView3(queryString1);
            break;
        case 3:
            string queryString2 = "SELECT Код_услуги,
Код_клиента,Тип_услуги, Дата_заказа, Дата_прибытия, Город_отправки, Город_прибытия,
Расстояние, Сумма, Код_тс FROM Услуги WHERE Услуги.Тип_услуги LIKE '" + textBox11.Text +
"%'";

            loadDataGridView3(queryString2);
            break;
        case 6:
            string queryString3 = "SELECT Код_услуги,
Код_клиента,Тип_услуги, Дата_заказа, Дата_прибытия, Город_отправки, Город_прибытия,
Расстояние, Сумма, Код_тс FROM Услуги WHERE Услуги.Город_отправки LIKE '" +
textBox14.Text + "%'";

            loadDataGridView3(queryString3);
            break;
        case 7:
            string queryString4 = "SELECT Код_услуги,
Код_клиента,Тип_услуги, Дата_заказа, Дата_прибытия, Город_отправки, Город_прибытия,
Расстояние, Сумма, Код_тс FROM Услуги WHERE Услуги.Город_прибытия LIKE '" +
textBox15.Text + "%'";

            loadDataGridView3(queryString4);
            break;
        case 8:
            string queryString5 = "SELECT Код_услуги,
Код_клиента,Тип_услуги, Дата_заказа, Дата_прибытия, Город_отправки, Город_прибытия,
Расстояние, Сумма, Код_тс FROM Услуги WHERE Услуги.Расстояние LIKE '" + textBox16.Text +
"%'";

            loadDataGridView3(queryString5);
            break;
        case 9:
            string queryString6 = "SELECT Код_услуги,
Код_клиента,Тип_услуги, Дата_заказа, Дата_прибытия, Город_отправки, Город_прибытия,
Расстояние, Сумма, Код_тс FROM Услуги WHERE Услуги.Сумма LIKE '" + textBox17.Text + "%'";
            loadDataGridView3(queryString6);
            break;
        case 10:
            string queryString7 = "SELECT Код_услуги,
Код_клиента,Тип_услуги, Дата_заказа, Дата_прибытия, Город_отправки, Город_прибытия,
Расстояние, Сумма, Код_тс FROM Услуги WHERE Услуги.Код_тс LIKE '" + textBox18.Text +
"%'";

```



```

        loadDataGridView3(queryString7);
        break;

    }
}
else if (counter == 0)
    MessageBox.Show("Заполните одно поле");
else
    MessageBox.Show("Заполните ТОЛЬКО одно поле");
}

private void button12_Click(object sender, EventArgs e)//Кнопка отмены для услуг
{
    string queryString1 = "SELECT Код_услуги, Код_клиента,Тип_услуги,
Дата_заказа, Дата_прибытия, Город_отправки, Город_прибытия, Расстояние, Сумма, Код_тс
FROM Услуги";
    loadDataGridView3(queryString1);
}

private void button13_Click(object sender, EventArgs e)//Выход
{
    Application.Exit();
}
private void button14_Click(object sender, EventArgs e)
{
    string tip = "SELECT Тип_услуги, COUNT(*) AS сум1 FROM Услуги GROUP BY
Тип_услуги ";

    OleDbCommand pre = new OleDbCommand();
    pre.CommandText = tip;
    pre.Connection = database;
    System.Data.DataTable dt = new System.Data.DataTable();
    OleDbDataAdapter adp = new OleDbDataAdapter(pre);

    adp.Fill(dt);

    chart1.DataSource = dt;
    chart1.Series["Тип"].XValueMember = "Тип_услуги";
    chart1.Series["Тип"].YValueMembers = "сум1";
    chart1.DataBind();
}

private void button15_Click(object sender, EventArgs e)
{
    string summa = "SELECT Тип_услуги, SUM( Сумма) AS сум2 FROM Услуги GROUP
BY Тип_услуги ";
    OleDbCommand pred = new OleDbCommand();
    pred.CommandText = summa;
    pred.Connection = database;
    System.Data.DataTable dt = new System.Data.DataTable();
    OleDbDataAdapter adp = new OleDbDataAdapter(pred);
    adp.Fill(dt);
    chart2.DataSource = dt;
    chart2.Series["Сумма"].XValueMember = "Тип_услуги";
    chart2.Series["Сумма"].YValueMembers = "сум2";
    chart2.DataBind();
}
}
}

```

СПИСОК ЛИТЕРАТУРЫ

1. Черушева Т.В. Проектирование программного обеспечения: учебное пособие / Т.В.Черушева. – Пенза: Издательство ПГУ, 2014. –170с.
2. Работа с базами данных на языке C#. Технология ADO .NET: уч. пособие / сост. О. Н. Евсеева, А.Б. Шамшев. –Ульяновск: УлГТУ, 2009. – 170с.
3. Лабор В.В. С шарп: создание приложений для windows/ В.В. Лабор – Ми.:Харвест, 2003.-384 с.