**СОДЕРЖАНИЕ**

[Введение 4](#_Toc73230604)

[1 Постановка задачи 5](#_Toc73230605)

[2 Анализ существующих решений 9](#_Toc73230606)

[3 Выбор и обоснование среды разработки 16](#_Toc73230607)

[4 Проектирование функциональной модели и базы данных 19](#_Toc73230608)

[4.1 Проектирование базы данных 19](#_Toc73230609)

[4.2 Логическое проектирование базы данных 19](#_Toc73230610)

[4.3 Физическое проектирование базы данных 23](#_Toc73230611)

[5 Проектирование интерфейса 26](#_Toc73230612)

[6 Реализация программного средства 33](#_Toc73230613)

[7 Тестирование и отладка 37](#_Toc73230614)

[8 Руководство пользователя 41](#_Toc73230615)

[Заключение 51](#_Toc73230616)

[Список использованных источников 52](#_Toc73230617)

[Приложение А 54](#_Toc73230618)

[Приложение Б 64](#_Toc73230619)

# Введение

Сегодня в начале 21 века, компьютеры можно встретить почти во всех сферах деятельности. Они достаточно прочно укрепились в нашей жизни и сознании. В настоящее время происходит стремительное развитие вычислительной техники и программного обеспечения ЭВМ.

Все большему числу специалистов в самых различных областях требуется доступ к вычислительной технике для решения стоящих перед ними задач. Сейчас компьютеры превратились в универсальные средства для обработки всех видов информации, используемых человеком. С помощью компьютеров не только проводятся числовые расчеты, но и подготавливаются к печати книги, создаются рисунки, кинофильмы, музыка, осуществляется управление заводами и космическими кораблями, проводятся всевозможные тестирования, проверяющие знания людей в той или иной области и так далее.

Одной из значительных ниш, которую заняли ЭВМ, является автоматизация документооборота и хранение больших объемов данных. Информационные технологии и развитие возможностей самих компьютеров позволяет хранить огромное количество данных, обеспечивая при этом высокую (в некоторых случаях почти мгновенную) скорость доступа к информации, возможность корректировки и модификации, переносимости и множимости данных, а если требуется, то их защиту. Также работа с базой данных на ЭВМ обычно более проста и удобна, чем работами с бумажными и другими носителями информации [1].

Все вышеперечисленные возможности реализуются с помощью систем управления базами данных. Сфера применения СУБД практически безгранична. трудно представить себе область интересов человека, которая не требовала бы хранения информации для дальнейшего использования.

Целью курсовой работы является написания программного средства для логистической компании.

# 1 Постановка задачи

Создание любой программы начинается с постановки задачи. Изначально задача формулируется в терминах предметной области, и необходимо перевести ее на язык понятий, более близких к программированию [2].

Поскольку программист редко досконально разбирается в предметной области, а заказчик – в программировании (простой пример: требуется написать бухгалтерскую программу), постановка задачи может стать весьма непростым итерационным процессом.

Кроме того, при постановке задачи заказчик зачастую не может четко и полно сформулировать свои требования и критерии.

Программная система должна обеспечивать хранение сведений о:

* автомобилях,
* грузоподъёмности транспорта,
* водителях
* заказах.

Предприятие владеет парком трейлеров. Каждый трейлер характеризуется:

* номерным знаком,
* маркой,
* фирмой,
* грузоподъёмностью,
* расходом топлива на 100 км,
* длиной прицепа,
* стоимостью перевозки (руб/км).

Имеются также водители:

* паспортные данные,
* стаж работы,
* классность,
* напарник (описывается так же, как водитель).

Предприятие получает заказы на перевозку товаров:

* вес,
* количество мест,
* пункт отправления,
* пункт назначения,
* расстояние (км),
* трейлер,
* водитель.

При расстоянии перевозки до 500 км водитель едет без напарника.

Должны быть созданы обобщенные списки:

* Сведения об автомобилях.
* Сведения о водителях и напарниках.
* Сведения о заказах.

Руководителю могут потребоваться следующие сведения:

* Список заказов на перевозку с сортировкой по расстоянию.
* Список заказов с сортировкой по весу.
* 5 наиболее большегрузных транспортов.
* Трейлер с наибольшим выполненным количеством заказов.
* Водитель с наибольшим выполненным количеством заказов.
* Рейсы, выполняемые без напарника.
* Общая прибыль предприятия.

Должна быть предусмотрена возможность добавления, удаления и изменения заказов.

Основные функции представлены в виде UML диаграммы вариантов использования.

При разработке диаграммы вариантов использования проектируемая система представляется в виде множества сущностей или актеров, взаимодействующих с системой с помощью вариантов использования. Актером называется любая сущность, взаимодействующая с системой извне . Вариант использования служит для описания сервисов, которые система предоставляет актеру. Диаграмма вариантов использования может дополняться пояснительным текстом, который раскрывает смысл или семантику составляющих ее компонентов [3].

Диаграмма вариантов использования представлена на рисунке 1.1.

В разрабатываемой системе возможна работа одного типа пользователя, соответственно на диаграмме вариантов использования представлен один актер, который обладает всеми функциями приложения, что позволяет ему взаимодействовать с данными всех таблиц приложения.

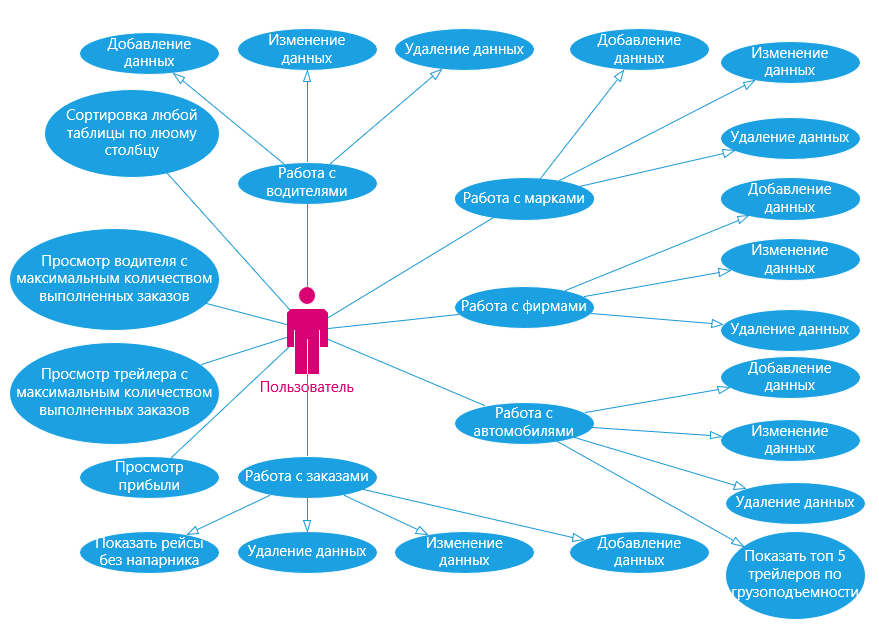


Рисунок 1.1 – Диаграмма вариантов использования

Для полноценной работы системы компьютер пользователя должен удовлетворять следующим аппаратным требованиям:

* процессор AMD Athlon X4 840 3.10 GHz;
* ОЗУ: не менее 4Гб;
* HDD: не менее 250Гб;
* видеоадаптер: DirectX9;
* видеопамять: не менее 512 МБ;

Кроме того, необходимы следующие программные компоненты:

* Windows 7 и выше;
* Microsoft Access для хранения базы данных и драйвера к нему;
* MS .NET Framework 4.5.2.

# 2 Анализ существующих решений

**Maxoptra**

Максоптра - онлайн-система управления логистикой, позволяющая автоматически распределить задачи между исполнителями и спланировать наиболее быстрые маршруты без лишних затрат. Поддерживает интеграцию с системами ГЛОНАСС и GPS для точного отслеживания доставки и отсутствия незапланированных задержек [4].

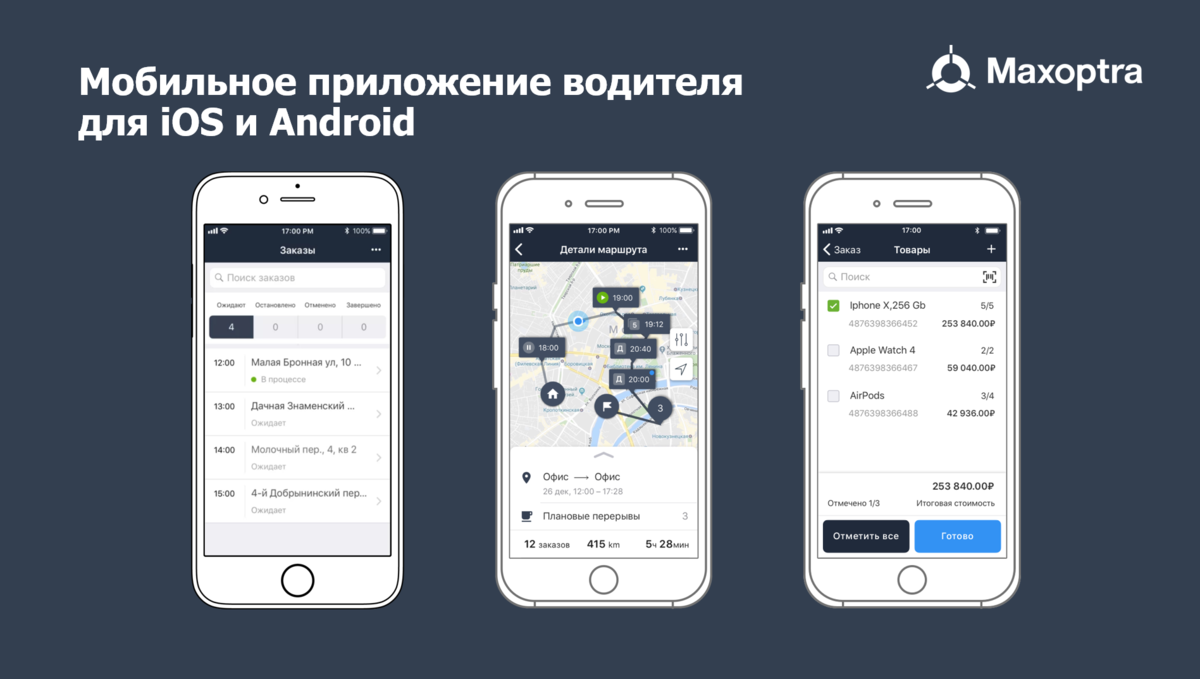


Рисунок 1.2 – Maxoptra

Онлайн-сервис включает в себя от инструментов контроля качества доставки до экрана планирования.

Оптимальное распределение заказов между исполнителями в автоматическом режиме, ручная корректировка маршрутов и добавление новых заявок в расписание.

Гибкая техническая архитектура позволяет эффективно управлять процессами в дистрибуции, торговом предприятии, оптовой и курьерской доставке, ритейле, интернет-магазине, мебельном и оконном бизнесе, службе доставки воды и даже в финансовом секторе.

В состав сервиса входят мобильное приложение для водителя на базе iOS / Android , веб - модули для диспетчеров и менеджеров колл - центра , модуль контроля доставки , система отчетности и аналитики .

* Автоматическое распределение заказов между курьерами
* Планирование оптимальных маршрутов с учетом пробок и других ограничений
* Прогноз ожидаемого времени прибытия с учетом информации о пробках , поступающих из Google
* Онлайн - мониторинг доставки в режиме “ план - факт ”
* Виджет клиента для самостоятельного отслеживания доставки
* Автоматические SMS / Email оповещения о статусе заказа для клиента
* Фотофиксация этапов выполнения работ
* Сбор электронных подписей получателей заказов
* Виджет для сбора отзывов клиентов о доставке

Тариф: от 1300 рублей - 2500 рублей + НДС за 1 автомобиль.14 дней пробного периода, бесплатный тариф отсутствует.

**КиберЛог**

Облачный сервис для управления транспортными перевозками, упрощающий взаимодействие между участниками бизнес процессов. Программа для транспортных компаний и владельцев транспорта [5].

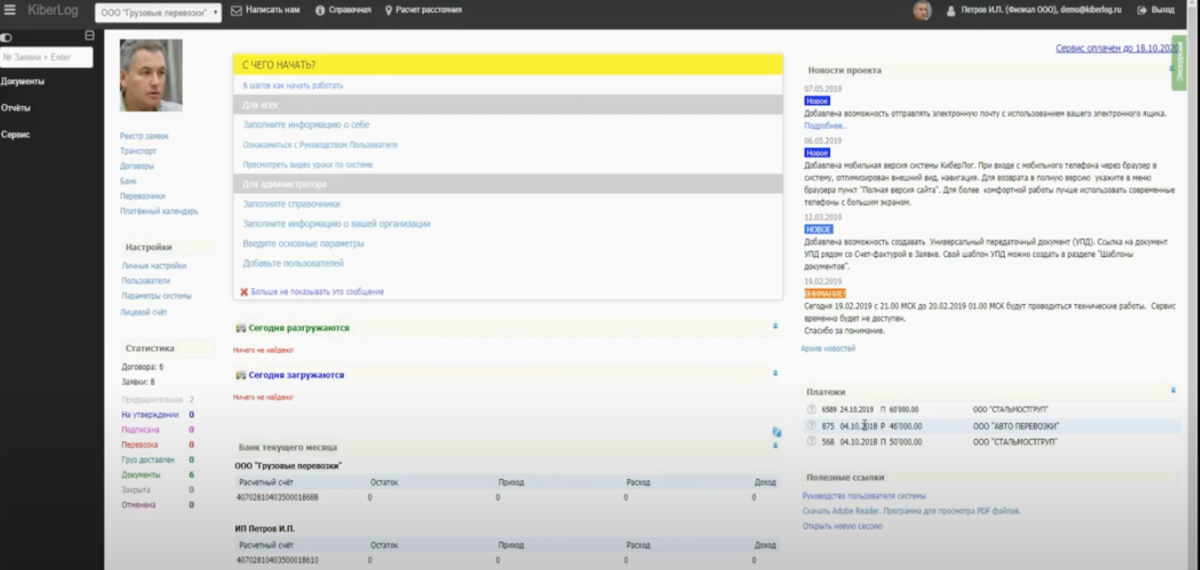


Рисунок 1.3 - КиберЛог

Отслеживание всех этапов выполнения заявки – от заключения договора и формирования оферты до доставки груза.

* Автоматическая архивация данных
* Передача данных по протоколу SSL
* Возможность добавить печать и подпись
* Интеграция с интернет-банками
* Сохранение доступа в систему после окончания бесплатного тестового периода

Тарифы: от 400 рублей, 30 дней пробного периода, бесплатный доступ отсутствует.

**ЯКурьер**

CRM-система для оптимизации транспортного отдела [6].

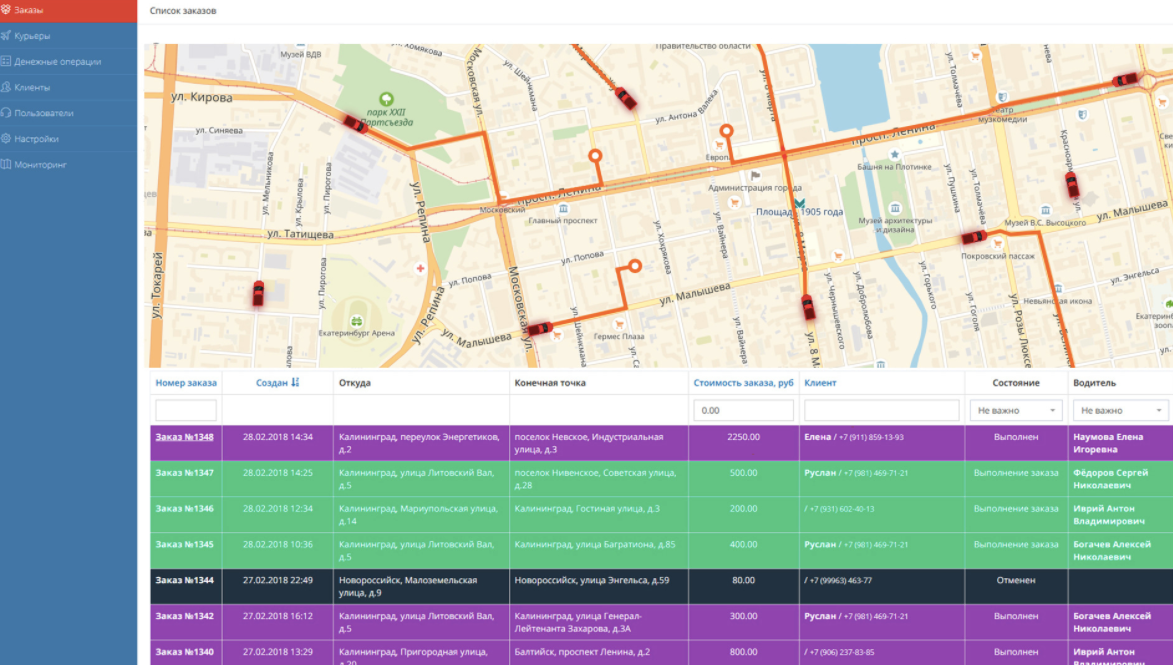


Рисунок 1.4 - ЯКурьер

Алгоритм построит оптимальный маршрут в соответствии весом, объемом и интервалом доставки. Водитель получит данный маршрут, контактные данные и комментарии к заказу.

* Агрегатор доставок/биржа грузоперевозок
* Оптимизация маршрутной доставки
* Мониторинг транспорта 2.0
* Таможенные процедуры

Тарифы: от 500 - 1000 рублей/месяц, 7 дней пробного периода, в бесплатном доступе для физ. лиц

**Мегалогист**

Разработан на платформе 1С:Предприятие 8 и предназначен для комплексной автоматизации транспортной логистики [7].

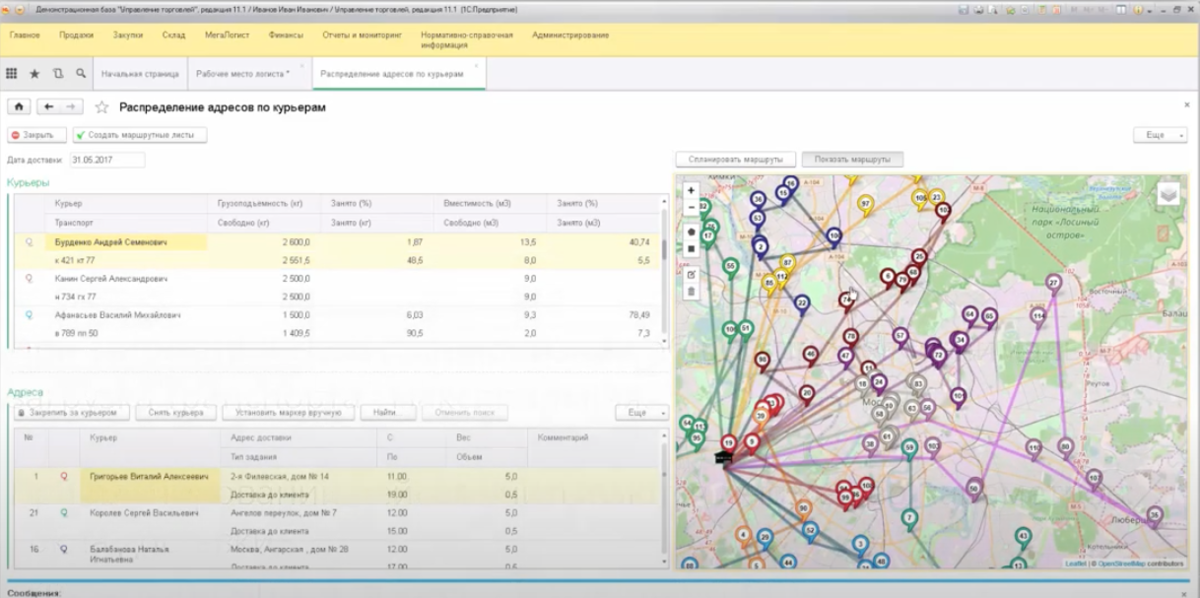


Рисунок 1.5 - Мегалогист

Создание заданий и планирование маршрутов, мониторинг рейсов, анализа KPI и рентабельности доставки.

* Единовременная оплата лицензии
* Планирование маршрутов в ручном и автоматическом режиме
* План/факт анализ
* Интеграция с онлайн-кассами (54-ФЗ)
* Внедрение и адаптация под бизнес-процессы клиента
* Бесплатный демо-доступ, удаленная демонстрация
* Диспетчеризация и мониторинг транспорта

Тарифы: цена по запросу. Возможен пробный период, отсутствие бесплатного тарифа.

**АвтоПеревозки**

Компания является ведущим разработчиком специализированного программного обеспечения для автоматизации бизнес-процессов и имеет сертификат и знак качества (Goods & Services Quality Ural Voluntary Certification Center) [8].

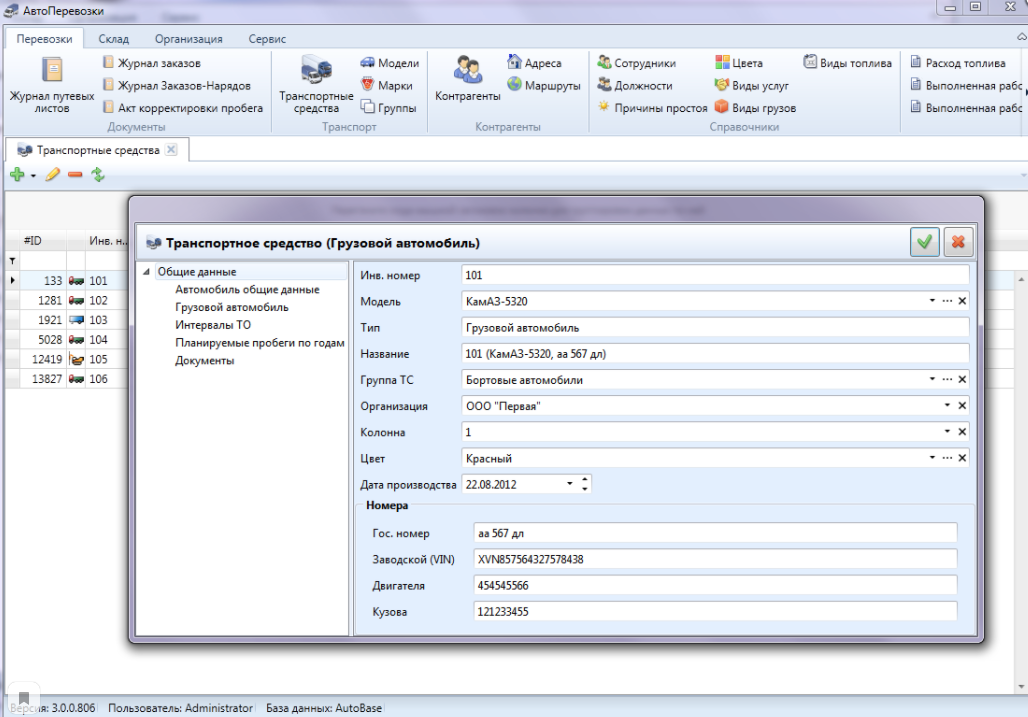


Рисунок 1.6 - АвтоПеревозки

Программные продукты AutoSoft помогают упорядочить, привести к единым стандартам и полностью автоматизировать учёт, обеспечить достоверную подготовку отчётности с учётом всех необходимых требований процесса ремонта и действующего законодательства.

* Синхронизация с базами данных MS SQL, LocalDB при наличии установленного драйвера
* Доработка под требования заказчика
* Настройка безопасности на основе ролей
* Работа с OpenOffice
* Потоковое формирование и печать нескольких путевых листов

Тариф: цена по запросу. Бесплатный доступ и пробное тестирование отсутствует.

**1С TMS Логистика**

Программа для планирования и учета деятельности транспортных компаний, анализирующая многие процессы, включая документооборот и организацию мультимодальной перевозок [9].

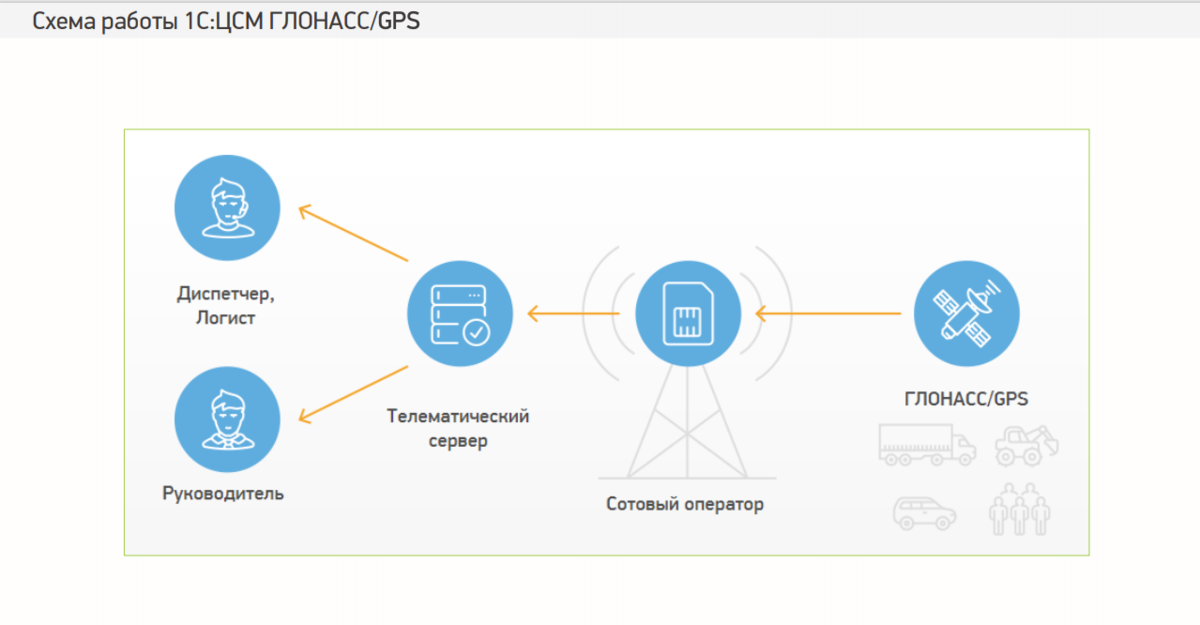


Рисунок 1.7 - 1С TMS Логистика

Программа будет полезна любым фирмам со службами доставки, поскольку позволяет планировать и контролировать

* ПО реализовано на платформе «1С: Предприятие 8»
* Интеграция в 1С ERP
* Возможность подключения дополнительных модулей
* Автоматическое планирование маршрутов
* Выгрузка заданий в АРМ экспедитора
* Формирование отчётов по доставкам план/факт

Тариф: цена по запросу. Бесплатный период и пробный период отсутствуют.

# 3 Выбор и обоснование среды разработки

Для разработки клиентского приложения была выбрана среда MicrosoftVisualStudio 2019. VisualStudio – это средство для разработки программного обеспечения, которое имеет большой функционал и возможности для отладки и развертывания приложений на различных платформах. В комплект разработчика входят интегрированная среда разработки (IDE) и инструменты тестирования для всех этапов процесса разработки программного обеспечения [10]. Интегрированная среда разработки представляет собой многофункциональную программу, которую можно использовать для различных аспектов разработки программного обеспечения.

MicrosoftVisualStudio поддерживает широкий спектр популярных языков программирования. Это хороший инструмент для профессиональных разработчиков программного обеспечения, которые хотят создавать коммерческие программы [11].

Основные преимущества MicrosoftVisualStudio:

- поддержка множества языков при разработке.

- редактор VisualStudio оптимизирует работу программиста по кодированию. В ходе того, как вводится новый код, на экране будут появляться подсказки. Для ускорения выполнения многих задач в VisualStudio предоставлено большое количество клавиатурных комбинаций. Существует набор средств быстрой переработки, которые позволяют быстро усовершенствовать структуру кода, не отрываясь от процесса программирования.

- интуитивный стиль кодирования. По умолчанию VisualStudio форматирует код по мере его ввода, автоматически вставляя необходимые отступы и применяя цветовое кодирование для выделения элементов типа комментариев. Такие незначительные отличия делают код более удобным для чтения и менее подверженным ошибкам.

- более высокая скорость разработки. Многие из функциональных возможностей VisualStudio направлены на то, чтобы помогать разработчику делать свою работу как можно быстрее. Удобные функции, вроде функции IntelliSense, функции поиска и замены и функции автоматического добавления и удаления комментариев позволяют разработчику работать быстро и эффективно.

- возможности отладки. Предлагаемые в VisualStudio инструменты отладки являются наилучшим средством для отслеживания ошибок и диагностирования странного поведения. Разработчик может выполнять свой код по строке за раз, устанавливать интеллектуальные точки прерывания, при желании сохраняя их для использования в будущем, и в любое время просматривать текущую информацию из памяти.

Для разработки приложения был выбран язык C#. C# предназначен для разработки приложений, которые выполняются в среде .NET Framework. Благодаря множеству нововведений C# обеспечивает возможность быстрой разработки приложений, но при этом сохраняет выразительность и элегантность, присущую языкам C. C# — это объектно-ориентированный язык, который поддерживает все правила объектно-ориентированного программирования: инкапсуляцию, наследование и полиморфизм. Язык поддерживает перегрузку операторов, указатели на функции-члены классов, атрибуты, события, свойства, исключения, комментарии в формате XML [6].

C# имеет множество функций, обеспечивающих надежность и устойчивость приложений. Сборка мусора автоматически освобождает память, занятую недостижимыми неиспользуемыми объектами. Обработка исключений предоставляет структурированный и расширяемый подход к обнаружению ошибок и их восстановлению. Типобезопасная структура языка делает невозможным чтение из неинициализированных переменных, индексацию массивов за пределами их границ или выполнение непроверенных приведений типов [12].

# 4 Проектирование функциональной модели и базы данных

## 4.1 Проектирование базы данных

База данных (БД) – совокупность структурированных данных, хранящихся в памяти электронной вычислительной машине.

Проектирование базы данных (БД) – процесс, который для заданного набора данных, относящихся к некоторой предметной области, позволяет выбрать и построить соответствующую оптимальную структуру [13].

При проектировании БД решаются три основные задачи:

− необходимость адекватного отображения предметной области и информационных потребностей пользователей в семантическую модель БД. Это инфологическое, или концептуальное проектирование БД;

− выбор способа отображения объектов предметной области в абстрактные объекты модели данных, чтобы это отображение не противоречило семантике предметной области и было по возможности эффективным. Это логическое проектирование БД;

− обеспечение эффективности выполнения запросов к БД, то есть каким образом расположить данные во внешней памяти, создать дополнительные структуры данных, например, индексы. Это физическое проектирование БД.

## 4.2 Логическое проектирование базы данных

Логическое проектирование – процесс конструирования информационной модели на основе выбора из существующих логических моделей данных, но не зависимой от конкретной системы управления базами данных и прочих физических условий реализации. Фаза логического проектирования состоит из двух этапов [14].

На первом этапе для каждой концептуальной модели данных, полученной на первой фазе проектирования, строится адекватная логическая модель, при этом выполняются следующие подзадачи:

– преобразование локальной концептуальной модели данных в локальную логическую модель, что предусматривает преобразование сложных связей и атрибутов;

– определение набора отношений исходя из структуры локальной логической модели;

– проверка модели с помощью правил нормализации;

– проверка модели в отношении транзакций пользователей;

– создание графического представления локальных моделей;

– проверка и обсуждение локальных логических моделей с конечными пользователями.

На втором этапе логического проектирования проводится создание и проверка глобальной логической модели данных на основе локальных логических моделей, при этом выполняются следующие подзадачи:

– слияние локальных логических моделей данных в единую глобальную логическую модель;

– проверка глобальной логической модели данных;

– проверка возможностей расширения модели в будущем;

– создание графического представления глобальной логической модели;

– проверка и обсуждение глобальной логической модели с конечными пользователями.

Концептуальное и логическое проектирования итерационные процессы, то есть они могут циклически продолжаться до тех пор, пока не закончится внесение уточнений и улучшений в составляемую схему БД.

На этапе логического проектирования разрабатывается логическая структура БД, соответствующая логической модели предметной области.

Проектирование схемы БД должно решать задачи минимизации дублирования данных и упрощения процедур их обработки и обновления.

В реляционных БД на данном этапе выполняются следующие действия:

– удаление связей M:N;

– рекурсивных связей;

– связей с атрибутами;

– множественных атрибутов;

– избыточных связей;

– перепроверка связей типа 1:1.

Далее выполняется нормализация отношений. Эдгаром Коддом был разработан аппарат нормализации отношений и был предложен механизм, позволяющий отношения преобразовать к третьей нормальной форме.

Нормализация отношений выполняется на основе анализа первичных ключей и существования функциональных зависимостей между атрибутами. Как правило нормализация выполняется в несколько этапов.

При проектировании реляционной БД требования первой нормальной формы должны выполняться всегда, остальные по желанию.

Первая нормальная форма:

– отсутствие повторяющихся записей;

– отсутствие повторяющихся групп полей;

– атрибуты и строки не упорядочены;

– значения атрибутов атомарны.

Вторая нормальная форма:

– удовлетворяет условия первой нормальной форме;

– любое неключевое поле однозначно идентифицируется полным набором ключевых полей.

Третья нормальная форма:

– удовлетворяет условия второй нормальной форме;

– ни одно из неключевых полей таблицы не идентифицируется с помощью другого неключевого поля.

База данных состоит из пяти сущностей: водителя, марок, фирм, трейлеров и заказов. Атрибуты каждой сущности показаны ниже.

Водитель

* ФИО
* Дата рождения
* Номер паспорта
* Стаж работы
* Классность

Марка

* Марка

Фирма

* Название фирмы

Трейлер

* Номерной знак
* Марка
* Фирма
* Грузоподъёмность
* Расходом топлива на 100 км
* Длина прицепа
* Стоимостью перевозки (руб/км)

Заказ

* Вес
* Количество мест
* Пункт отправления
* Пункт назначения
* Расстояние (км)
* Трейлер
* Водитель
* Дата создания заказа
* Дата выполнения заказа

## 4.3 Физическое проектирование базы данных

Физическое проектирование – процесс реализации БД, размещаемой во вторичной памяти вычислительной машины, при котором выполняется отображение созданной глобальной логической модели на особенности конкретной системы управления базами данных.

Для реляционных систем управления базами данных процесс можно разбить на этапы и подзадачи. Каждую подзадачу легко распределить между разработчиками базы данных [15].

В первом этапе физического проектирования БД осуществляется перенос глобальной логической модели в среду целевой системы управления базами данных. Выполняется проектирование базовых таблиц и реализация бизнес-правил.

Второй этап включает в себя проектирование физического представления БД. На этом этапе происходит анализ транзакций, выбор файловой структуры, определение вторичных индексов, анализ необходимости введения контролируемой избыточности данных и определение требований к дисковой памяти.

В процессе третьего этапа разрабатываются механизмы защиты, заключающиеся в разработке пользовательских представлений и определение прав доступа.

На этапе четвертом организуется мониторинг и осуществляется настройка функционирования системы.

Общая методология позволяет повысить общую эффективность работы по проектированию БД благодаря циклическому процесс проектирования, а также существенная роль отводится конечным пользователям, привлекаемым для ознакомления и проверки модели данных и проектной документации. Выполнение той или иной подзадачи в методологии не регламентировано жестко, и порядок их выполнения в рамках этапов может меняться по усмотрению разработчиков.

Этап физического проектирования заключается в увязке логической структуры БД и физической среды хранения с целью наиболее эффективного размещения данных. Решается вопрос размещения хранимых данных в пространстве памяти, выбора эффективных методов доступа к различным компонентам физической БД. Принятые на этом этапе решения оказывают определяющее влияние на производительность системы.

Использование реляционной базы данных для хранения объектно-ориентированных данных приводит к семантическому разрыву, заставляя программистов писать программное обеспечение, которое должно уметь как обрабатывать данные в объектно-ориентированном виде, так и уметь сохранить эти данные в реляционной форме.

Эта постоянная необходимость в преобразовании между двумя разными формами данных не только сильно снижает производительность, но и создает трудности для программистов, так как обе формы данных накладывают ограничения друг на друга. Поэтому используют ORM библиотеки.

Полученная при физическом проектировании схема базы данных отображена на рисунке 4.1.

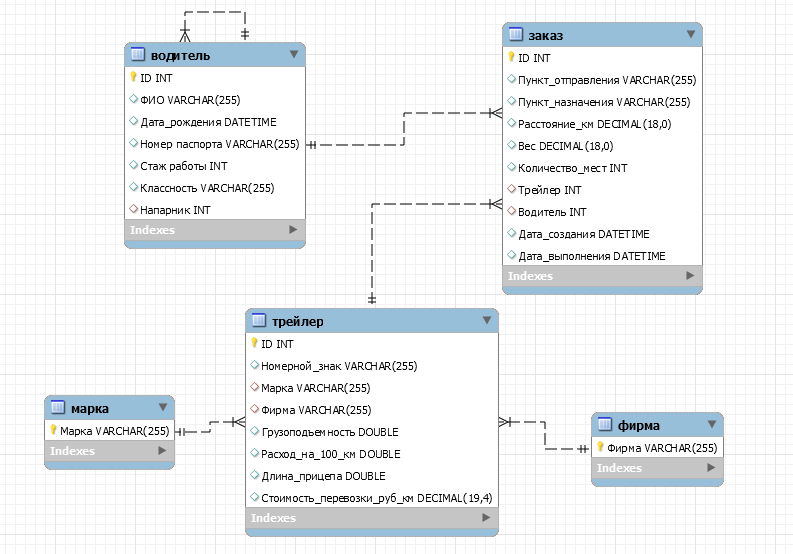


Рисунок 4.1 – Схема базы данных

# 5 Проектирование интерфейса

Пользовательский интерфейс должен характеризоваться удобством, эффективностью, понятностью. Хороший интерфейс предоставляет пользователю наиболее удобный способ взаимодействия с программным обеспечением путем обеспечения логичности и простоты в расположении элементов управления.

Windows Forms - интерфейс программирования приложений (API), отвечающий за графический интерфейс пользователя и являющийся частью Microsoft .NET Framework. Данный интерфейс упрощает доступ к элементам интерфейса Microsoft Windows за счет создания обёртки для существующего Win32 API в управляемом коде. Причём управляемый код - классы, реализующие API для Windows Forms, не зависят от языка разработки. То есть программист одинаково может использовать Windows Forms как при написании ПО на C++, С#, так и на VB.Net, J# и др [16].

Приложение Windows Forms представляет собой событийно-ориентированное приложение, поддерживаемое Microsoft .NET Framework. В отличие от пакетных программ, большая часть времени тратится на ожидание от пользователя каких-либо действий, как, например, ввод текста в текстовое поле или клика мышкой по кнопке.

В Windows Forms форма - это видимая поверхность, на которой выводится информация для пользователя. Обычно приложение Windows Forms строится путем помещения элементов управления на форму и написания кода для реагирования на действия пользователя, такие как щелчки мыши или нажатия клавиш. Элемент управления - это отдельный элемент пользовательского интерфейса, предназначенный для отображения или ввода данных.

При выполнении пользователем какого-либо действия с формой или одним из ее элементов управления создается событие. Приложение реагирует на эти события с помощью кода и обрабатывает события при их возникновении.

Интерфейс сделан в минималистичной форме и содержит только самые необходимые инструменты, чтобы не перегружать пользователя информацией.

При открытии приложения, пользователю будет показана главная форма приложения, на которой расположены кнопки перехода к другим формам.

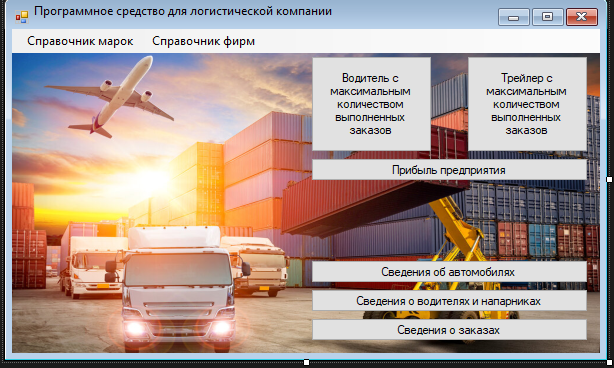


Рисунок 5.1 – Главная форма

При переходе на справочник марок, пользователю будет отображена форма для работы с марками. На данной форме пользователь может добавить информацию в таблицу, изменить уже существующую иди удалить строку вовсе.

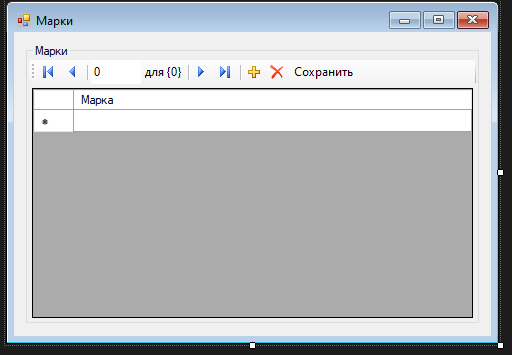


Рисунок 5.2 – Форма работы с марками

Стоить отметить, что каждая форма поддерживает изменение размера окна, т.е., к примеру, при растяжении форма на весь экран, все кнопки или поля буду отображены в своих углах.

При переходе на форму справочника фирм, будет отображена форма для работы с фирмами. На данной форме пользователь может добавить информацию в таблицу, изменить уже существующую иди удалить строку вовсе.

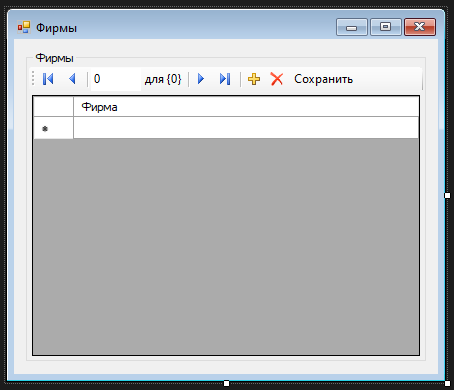


Рисунок 5.3 – Форма работы с фирмами

При переходе на форму водителей, пользователю будет отображена форма для работы с водителями. На данной форме пользователь может добавить информацию в таблицу, изменить уже существующую иди удалить строку вовсе.

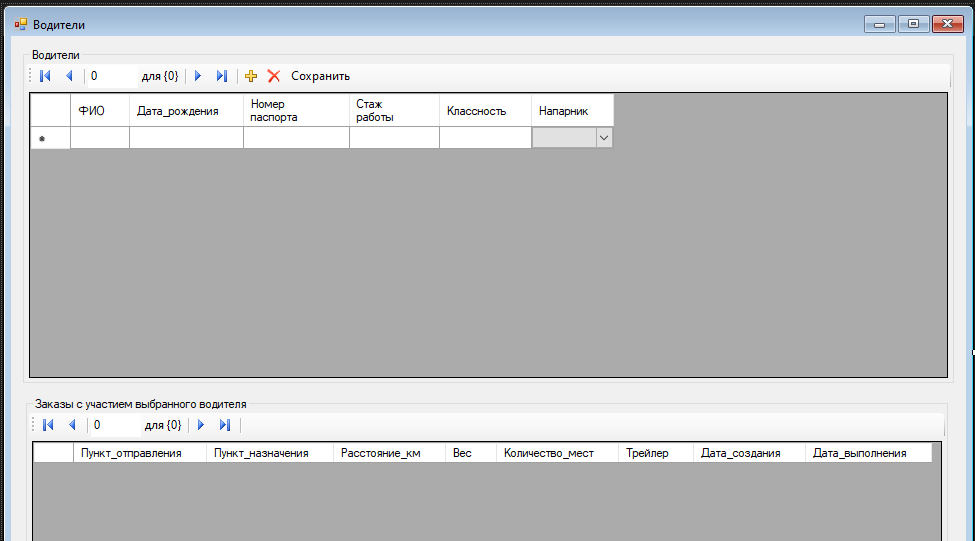


Рисунок 5.4 – Форма работы с водителями

Стоит сказать, что на форме водителей при выборе любого из списка, в таблице ниже будет отображена информация о рейсах, в которых участвовал данный водитель.

При выборе меню транспорта, пользователю будет отображена форма для работы с трейлерами. На данной форме пользователь может добавить информацию в таблицу, изменить уже существующую иди удалить строку вовсе.

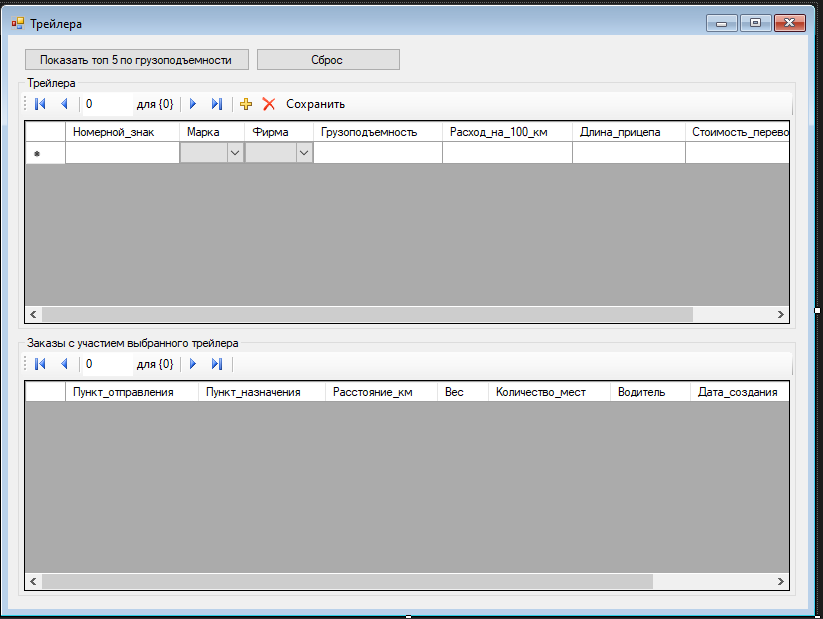


Рисунок 5.5 – Форма работы с трейлерами

На данной форме, как и на форме водителей, при выборе любого транспортного средства из списка, пользователю будет отображен в таблице ниже список всех заказов, в которых участвовал данный трейлер. На данной форме пользователь также может посмотреть список из пяти самых грузоподъемных машин.

При выборе пункта заказов, пользователю будет отображена форма для работы с заказами. На данной форме пользователь может добавить информацию в таблицу, изменить уже существующую иди удалить строку вовсе.

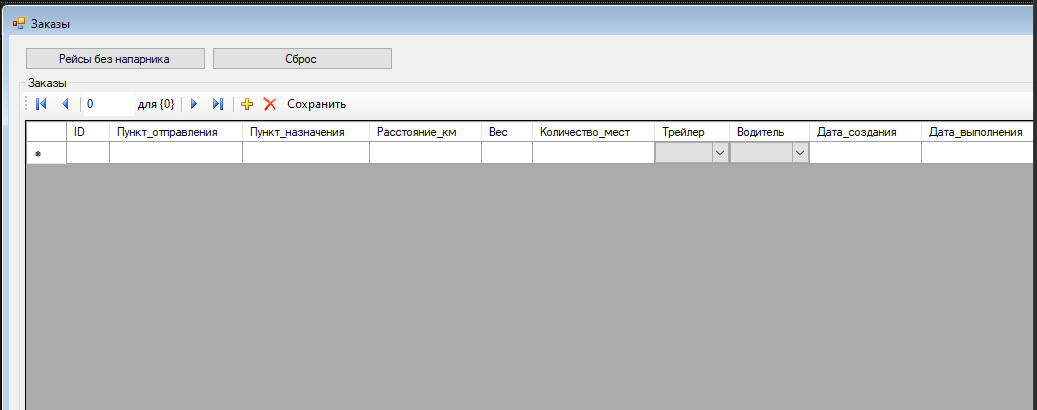


Рисунок 5.6 – Форма работы с заказами

Также стоит отметить, что на данной форме пользователь может отдельно посмотреть все рейсы, которые совершаются без напарников.

При выборе пункта просмотра лучшего водителя, будет отображена форма для показа ФИО водителя и количество выполненных им заказов.

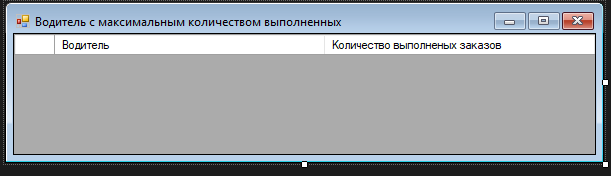


Рисунок 5.7 – Форма вывода водителя с максимальным количеством выполненных заказов

При выборе пункта с самым используемым трейлеором будет отображена форма показа номера машины и количества выполненных на ней заказов.

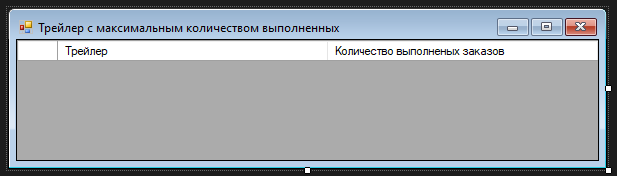


Рисунок 5.8 – Форма вывода трейлера с максимальным количеством выполненных заказов

При выборе пункта общей прибыли, пользователю будет отображена форма с детальной суммой по каждому товару, а также кнопка на ней для подсчета общей суммы по всем заказам.

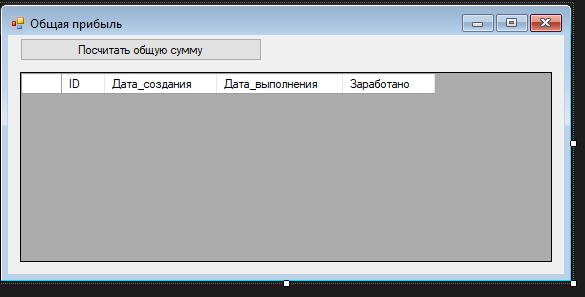


Рисунок 5.9 – Форма вывода прибыли по заказам

На данном моменте этап проектирования приложения окончен.

# 6 Реализация программного средства

Приложение будет построено на базе архитектуры MVC.

Термин модель-представление-контроллер (model-view-controller) используется с конца 70-х гг. прошлого столетия. Этот шаблон происходит из проекта Smalltalk, выполнявшегося в Xerox PARC, где он был задуман как способ организации ранних приложений с графическим пользовательским интерфейсом. Некоторые нюансы первоначального шаблона MVC были связаны с концепциями, специфичными для Smalltalk, такими как экраны и инструменты, но более широкие понятия по-прежнему применимы к приложениям - и особенно хорошо они подходят для веб-приложений [17].

Взаимодействие с приложением MVC осуществляется в соответствии с естественным циклом действий пользователя и обновлений представления, при котором предполагается, что представление не содержит информации о состоянии. Это прекрасно сочетается с запросами и ответами HTTP, которые лежат в основе веб-приложения.

Более того, инфраструктура MVC принудительно применяет разделение ответственности - модель предметной области и логика контроллера отделены от пользовательского интерфейса. В веб-приложении это означает, что HTML-разметка хранится отдельно от остальной части приложения, что упрощает и облегчает сопровождение и тестирование.

Появление платформы Ruby on Rails привело к возобновлению всеобщего интереса к MVC, и он остается шаблоном реализации архитектуры MVC. С тех пор появилось много других инфраструктур MVC, и все они демонстрировали преимущества архитектуры MVC - разумеется, это относится и к ASP.NET MVC.

Если оперировать высокоуровневыми понятиями, то архитектурный шаблон MVC означает, что приложение MVC будет разделено, по крайней мере, на три части, которые описаны ниже:

Модели, содержащие или представляющие данные, с которыми работают пользователи. Они могут быть простыми моделями представлений, которые только представляют данные, передаваемые между представлениями и контроллерами, или же они могут быть моделями предметной области, которые содержат бизнес-данные, а также операции, преобразования и правила для манипулирования этими данными.

Представления, применяемые для визуализации некоторой части модели в виде пользовательского интерфейса.

Контроллеры, которые обрабатывают поступающие запросы, выполняют операции с моделью и выбирают представления для визуализации пользователю.

Модели - это определение "вселенной", в которой функционирует ваше приложение. Например, в банковском приложении модель представляет все аспекты банковской деятельности, поддерживаемые приложением, такие как расчетные счета, главная бухгалтерская книга и кредитные лимиты для клиентов, равно как и операции, которые могут использоваться для манипулирования данными в модели, такие как внесение денежных средств и списание со счетов. Модель отвечает также за сохранение общего состояния и целостности данных - например, удостоверяясь, что все транзакции внесены в главную книгу, а клиент не снимает со счета больше денежных средств, чем имеет на то право, или больше, чем имеется в распоряжении самого банка.

Модели определяются также и тем, за что они не отвечают: модели не имеют дела с визуализацией пользовательских интерфейсов или обработкой запросов, т.к. это ответственность представлений и контроллеров.

Представления содержат логику, необходимую для отображения элементов модели пользователю - и ничего более. Они не имеют никаких прямых сведений о модели и не обмениваются данными непосредственно с ней. Контроллеры являются шлюзом между представлениями и моделью - запросы поступают от клиента и обслуживаются контроллером, который выбирает подходящее представление для отображения пользователю и, если требуется, соответствующую операцию, которая должна быть выполнена над моделью.

Каждая часть архитектуры MVC является четко определенной и самодостаточной - это то, что называется разделением ответственности. Логика, которая манипулирует данными в модели, содержится только в модели, логика, отображающая данные - только в представлении, а код, который обрабатывает запросы и ввод пользователей - только в контроллере. При четком разграничении всех частей приложение будет легче сопровождать и расширять в течение его срока существования, независимо от того, насколько большим оно станет.

Каждой форме назначен свой класс для взаимодействия с ней.

Диаграммы классов являются одной из форм статического описания системы с точки зрения ее проектирования, показывая ее структуру. Диаграмма классов не отображает динамическое поведение объектов, изображенных на ней классов. На диаграммах классов показываются классы, интерфейсы и отношения между ними. Диаграмма классов может отражать, в частности, различные взаимосвязи между отдельными сущностями предметной области, такими как объекты и подсистемы, а также описывать их внутреннюю структуру и типы отношений. Каждый блок на диаграмме классов соответствует разработанному классу в программе. Атрибутами являются элементы, отображаемые на форме (поле, надпись, список), а также объявленные при необходимости переменные. Операции представляют собой функции и процедуры класса.

Диаграмма классов представлена на рисунке 6.1:

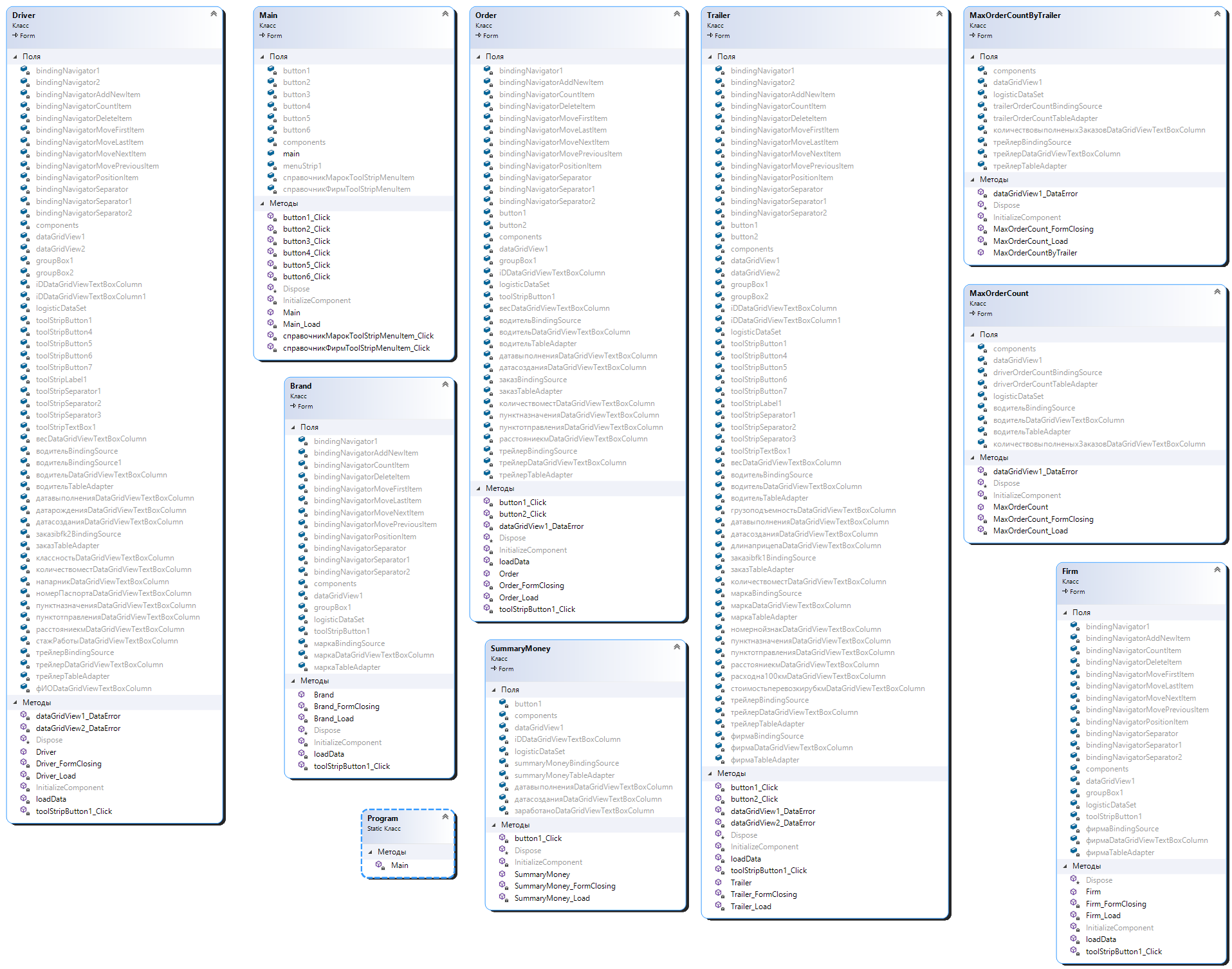


Рисунок 6.1 – Диаграмма классов

# 7 Тестирование и отладка

Тестирование программного обеспечения – процесс анализа программного средства и сопутствующей документации с целью выявления дефектов и повышения качества продукта.

Существующие на сегодня методы тестирования программ не позволяют однозначно и полностью выявить все дефекты и установить корректность функционирования анализируемой программы, поэтому все существующие методы тестирования действуют в рамках формального процесса проверки разрабатываемого мобильного приложения [18].

Поэтому можно выделить три основные цели тестирования:

− повысить вероятность того, что приложение, предназначенное для тестирования, будет работать правильно при любых обстоятельствах;

− повысить вероятность того, что приложение, предназначенное для тестирования, будет соответствовать всем описанным требованиям;

− предоставление актуальной информации о состоянии продукта на данный момент.

Виды тестирования:

− функциональное тестирование;

− системное тестирование;

− тестирование производительности

− регрессионное;

− модульное;

− тестирование безопасности;

− тестирование локализации;

− юзабилити.

При функциональное тестировании выполняется проверка соответствия программного обеспечения требованиям, заявленным в спецификации. Может проводиться как полное тестирование заявленной функциональность, так и проверка только базовой функциональности. В основном для функционального тестирования используются ручные тесты, которые легче адаптировать под нужные цели и задачи.

Функциональное тестирование программного обеспечения полностью имитирует фактическое использование системы и позволяет своевременно выявить системные ошибки программы и, тем самым, избежать множества проблем при работе с ним в дальнейшем.

Системное тестирование – высокоуровневая проверка функционала всей программы или системы в целом.

Тестирование производительности проводится с целью определения, как быстро работает система или её часть под определённой нагрузкой. Подразделяется на нагрузочное и стресс тестирования. Первое предназначено для проверки работоспособности системы при стандартных нагрузках и для определения максимально возможного пика, при котором система работает правильно, а второе – для проверки работоспособности системы при нестандартных нагрузках и для определения максимально возможного пика, при котором система работает правильно.

С целью проверить, не влияют ли новые функции, улучшения и исправленные дефекты на существующую функциональность продукта и не возникают ли старые дефекты проводится регрессионное тестирование.

Каждая сложная программная система состоит из отдельных частей - модулей, выполняющих ту или иную функцию в составе системы. Для того, чтобы удостовериться в корректной работе всей системы, необходимо вначале протестировать каждый модуль системы по отдельности, то есть выполняется модульное тестирование. В случае возникновения проблем при тестировании системы в целом это позволяет проще выявить модули, вызвавшие проблему, и устранить соответствующие дефекты в них.

Тестирование безопасности – это стратегия тестирования, используемая для проверки безопасности системы, а также для анализа рисков, связанных с обеспечением целостного подхода к защите приложения, атак хакеров, вирусов, несанкционированного доступа к конфиденциальным данным.

Тестирование локализации – это процесс тестирования локализованной версии программного продукта. Проверка правильности перевода элементов интерфейса пользователя, проверка правильности перевода системных сообщений и ошибок, проверка перевода и сопроводительной документации.

Для тестирования удобства пользования проводят такое вид тестирования, как юзабилити тестирование, направленный на установление степени удобства использования, обучаемости, понятности и привлекательности для пользователей разрабатываемого продукта в контексте заданных условий. Выявлять проблемы, связанные со специфическим механизмом интерфейса определять, существуют ли проблемы с удобностью интерфейса для навигации, использования основного функционала.

На основе функций, которые должны быть протестированы в разработанной информационной системе, составлен чек-лист (checklist), приведенный в таблице 7.1.

Таблица 7.1 – Чек-лист

|  |  |  |  |
| --- | --- | --- | --- |
| **Тестируемый модуль** | | **Тестируемая функция** | **Результат** |
| Запуск приложения | | Функция запуска | Выполнено успешно |
| Выход из приложения | | Функция закрытия программы |
| Ввод данных | | Определение соответствие типа вводимых данных | Выполнено успешно |
| Ввод ошибочных данных | Вывод сообщении об ошибке | | Выполнено успешно |
| Сохранение и загрузка | Сохранение данных в базу данных | | Выполнена успешно |
| Чтение данных из базы данных | |

В качестве тестирования программного продукта был выбран тест-кейс (**Test Case)**.

**Test Case** - это артефакт, описывающий совокупность шагов, конкретных условий и параметров, необходимых для проверки реализации тестируемой функции или её части.

Тестирование производилось на ОС «Windws 10.

Таблица 7.2 – Тест-кейсы

|  |  |  |
| --- | --- | --- |
| **Краткое описание** | **Шаги для воспроизведения** | **Ожидаемый результат** |
| Добавление новой записи в базу данных | Заполнены все столбцы строки, нажата кнопка Enter, нажата кнопка сохранить, сохранение подтверждено | Уведомление об успешном сохранении. Новые данные отображаются в общем списке |
| Редактирование конкретного кейса | Выбрана строка таблицы, изменены требуемые данные, нажата клавиша Enter, нажата клавиша сохранить и сохранение подтверждено» | Уведомление об успешном сохранении. Обновленные данные отображаются в таблице. |
| Удаление строки из бд | Нажата кнопка «Удалить», нажата клавиша сохранить и сохранение подтверждено» | Уведомление об успешном сохранении. Удаленные данные не отображаются в таблице. |
| Сортировка | Нажатие на название любого столбца | Данные отсортированы по столбцу |

# 8 Руководство пользователя

Первым делом, для работы приложения необходимо создать базу данных. Код создания базы данных указан в приложении Б.

При создании приложения использовалась база данных MySQL, с доступом по логину root и паролю 1234. Если пользователь при установке указывает другой логин или пароль, их нужно указать в файле App.config проекта, после чего пересобрать проект.

При открытии приложения, пользователю будет показана главная форма приложения, на которой расположены кнопки перехода к другим формам.

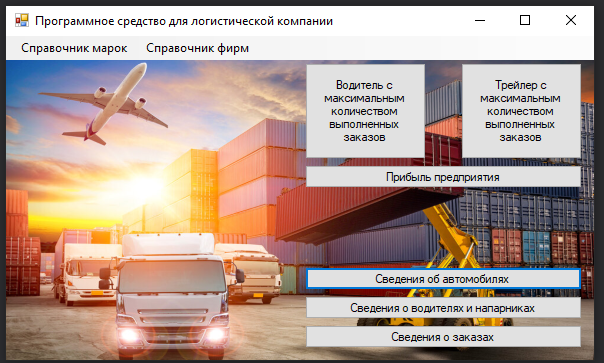


Рисунок 8.1 – Главная форма

При переходе на справочник марок, пользователю будет отображена форма для работы с марками. На данной форме пользователь может добавить информацию в таблицу, изменить уже существующую иди удалить строку вовсе.

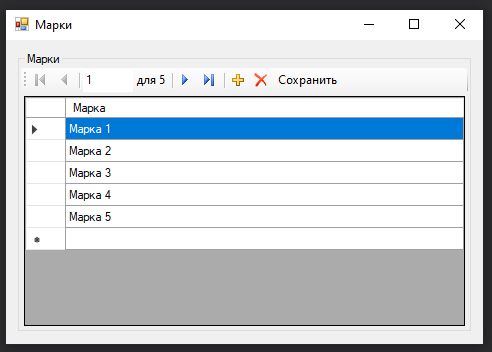


Рисунок 8.2 – Форма для работы с марками

Добавим новую марку. После внесения данных в последнюю строку необходимо сохранить данные в базе данных, для этого нужно нажать кнопку сохранения. Появится запрос на сохранение.

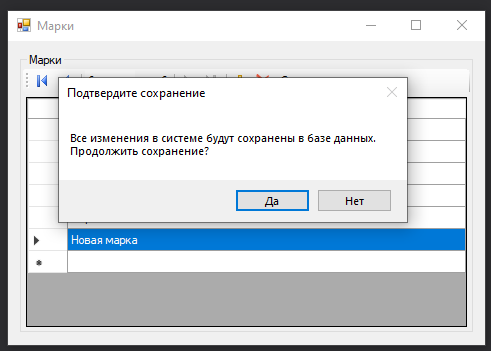


Рисунок 8.3 – Запрос на сохранение изменений

Сделано это для того, чтобы пользователь при случайном удалении данных из таблицы не терял их. После подтверждения сохранения будет отображен его результат.

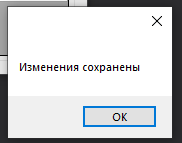


Рисунок 8.4 – Результат сохранения изменений

Все, марка сохранена в базе данных и ее можно использовать в дальнейшем.

При переходе на форму справочника фирм, будет отображена форма для работы с фирмами. На данной форме пользователь может добавить информацию в таблицу, изменить уже существующую иди удалить строку вовсе. Работа на этой форме идет точно по такому же принципу, что и на форме марок.

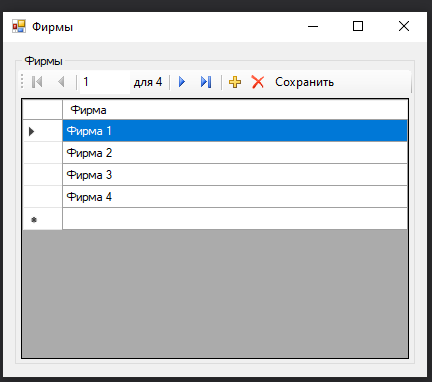


Рисунок 8.5 –Форма для работы с фирмами

При выборе меню транспорта, пользователю будет отображена форма для работы с трейлерами. На данной форме пользователь может добавить информацию в таблицу, изменить уже существующую иди удалить строку вовсе.

На данной форме, как и на форме водителей, при выборе любого транспортного средства из списка, пользователю будет отображен в таблице ниже список всех заказов, в которых участвовал данный трейлер.

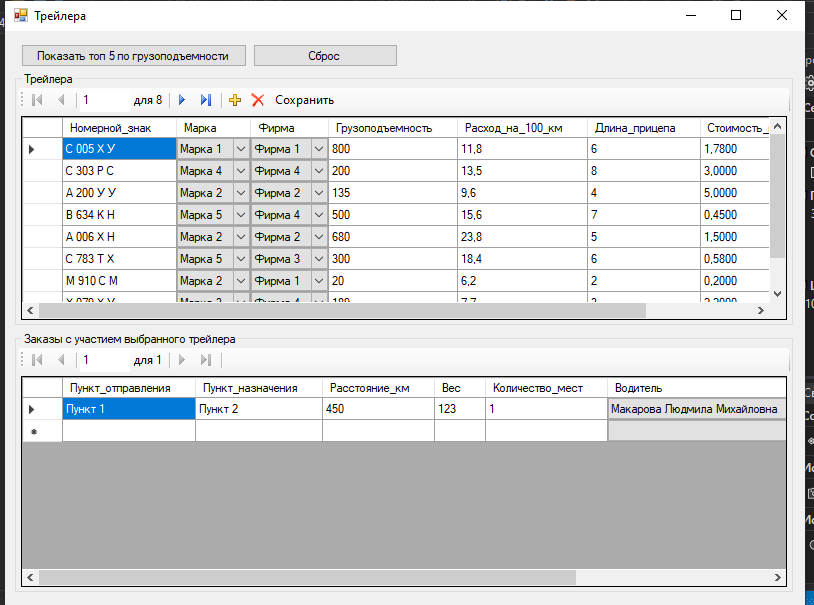


Рисунок 8.6 – Форма для работы с трейлерами

При добавлении нового транспортного средства видно, что в списке марок отображается созданная нами ранее марка.

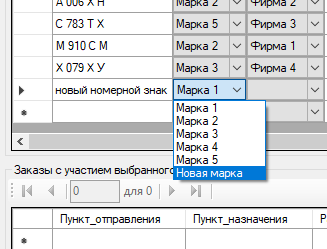


Рисунок 8.7 – Выбор добавленной ранее марки из выпадающего списка

Все таблицы поддерживают сортировку по любому столбцу.

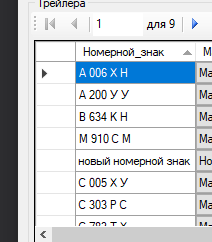


Рисунок 8.8 – Результат сортировки машин по номерному знаку

На данной форме пользователь также может посмотреть список из пяти самых грузоподъемных машин.

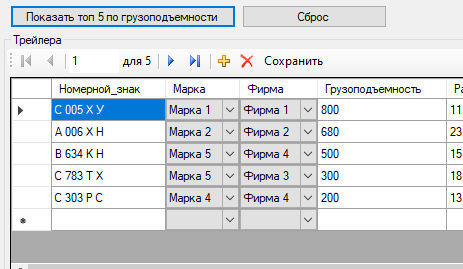


Рисунок 8.9 – Отображение топ 5 машин по грузоподъемности

При переходе на форму водителей, пользователю будет отображена форма для работы с водителями. На данной форме пользователь может добавить информацию в таблицу, изменить уже существующую иди удалить строку вовсе.

Стоит сказать, что на форме водителей при выборе любого из списка, в таблице ниже будет отображена информация о рейсах, в которых участвовал данный водитель.

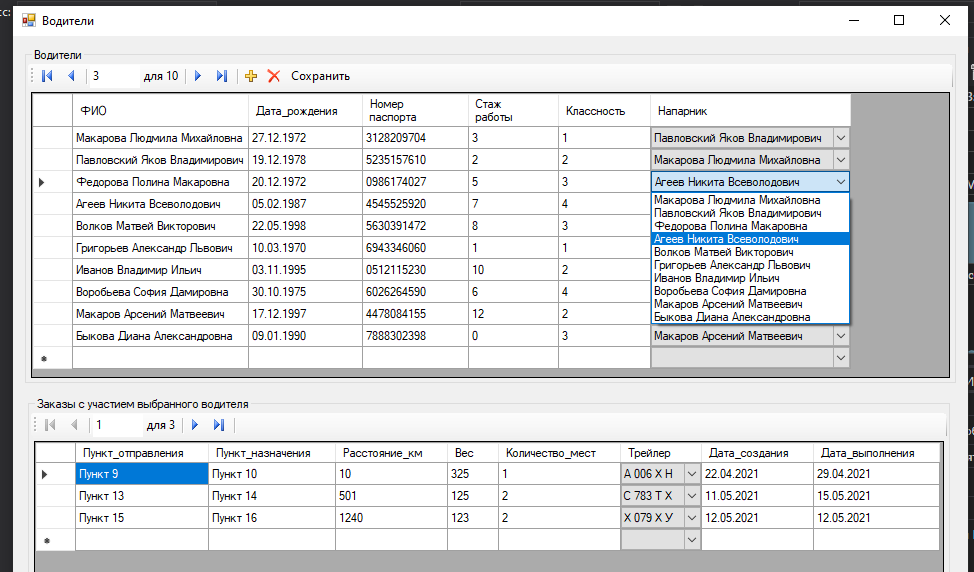


Рисунок 8.10 – Форма работы с водителями и их напарниками

При выборе пункта заказов, пользователю будет отображена форма для работы с заказами. На данной форме пользователь может добавить информацию в таблицу, изменить уже существующую иди удалить строку вовсе.

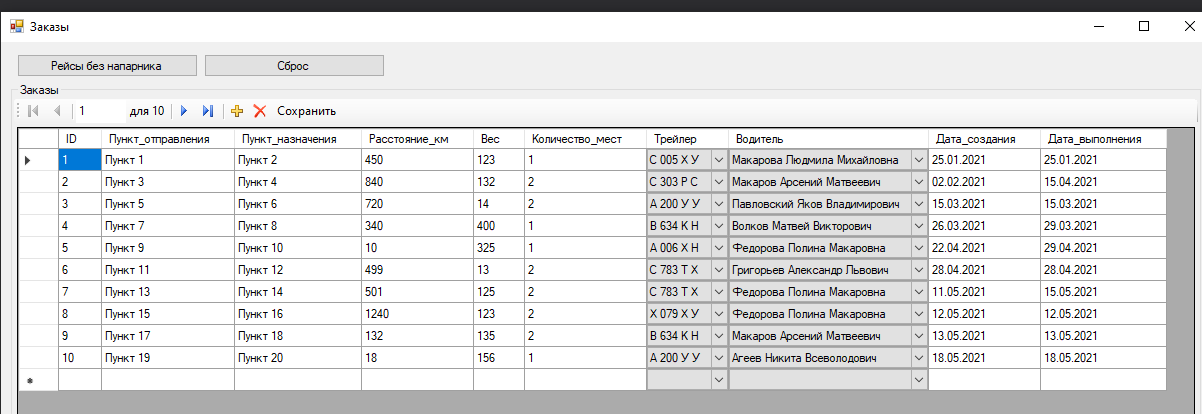


Рисунок 8.11 – Форма работы с заказами

При оформлении нового заказа можно заметить, что в списке трейлеров отображается номерной знак трейлера, добавленного нами ранее.

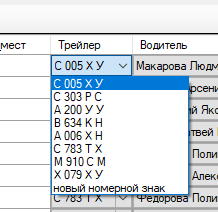


Рисунок 8.12 – Выбор добавленного ранее трейлера из выпадающего списка

По требованиям также было необходимо реализовать отображения рейсов без напарника, что также можно посмотреть на данной форме.

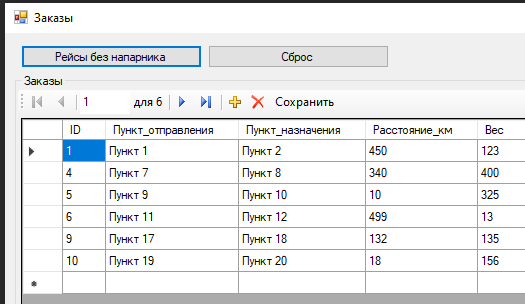


Рисунок 8.13 – Отображение рейсов без напарников

При выборе пункта просмотра лучшего водителя, будет отображена форма для показа ФИО водителя и количество выполненных им заказов.

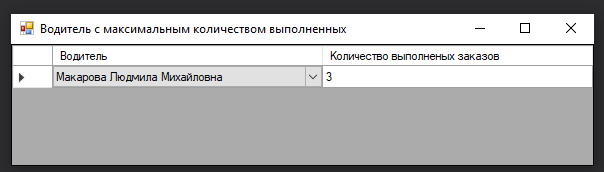


Рисунок 8.14 – Форма отображения водителя с максимальным количеством выполненных заказов

При выборе пункта с самым используемым трейлеором будет отображена форма показа номера машины и количества выполненных на ней заказов.

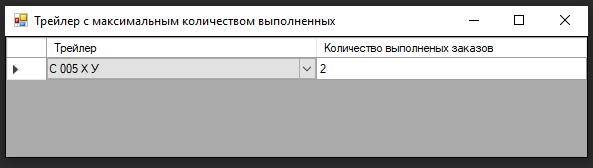


Рисунок 8.15 – Форма отображения номера трейлера с максимальным количеством выполненных заказов

При выборе пункта общей прибыли, пользователю будет отображена форма с детальной суммой по каждому товару, а также кнопка на ней для подсчета общей суммы по всем заказам.

Сумма заработка по заказу считается как количество километров в заказе \* цена за 1км выбранного в заказе трейлера.

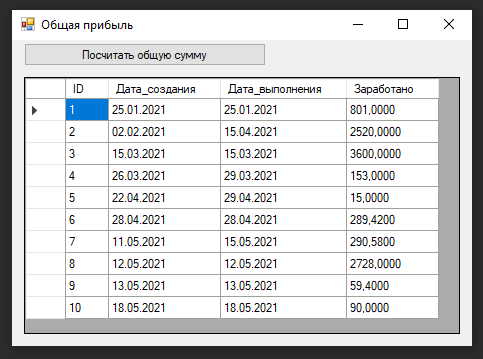


Рисунок 8.16 – Форма отображения общей прибыли

Общая сумма по всем заказам считается как сумма заработка по каждому заказу.

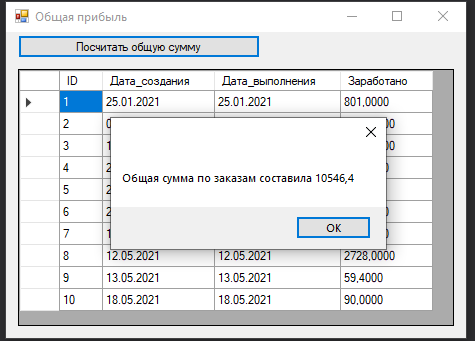


Рисунок 8.17 – Результат общей суммы по заказам

# Заключение

Главным итогом курсового проектирования является разработанное программное средство для логической компании, которое хранит данные о водителях, автомобилях, марках, фирмах и заказах.

В рамках данного курсового проекта была достигнута главная цель, которая заключалась в реализации базы данных и программного приложения для улогистической компании. Для ее достижения использовались стандартные компоненты среды Visual Studio и язык программирования высокого уровня C#.

Разработанное программное средство представляет собой систему, предназначенную для облегчения работы c базой данных по ведению заказов логистической компании.

Данная программа полностью соответствует требованиям технического задания. Программный продукт может эффективно функционировать на любом персональном компьютере, удовлетворяющем техническим требованиям

# Список использованных источников

1. Развитие информационных технологий [Электронный ресурс]. – Режим доступа: https://intuit.ru/studies/courses/4115/1230/lecture/24057. – Дата доступа: 30.05.2021.
2. Разработка внутренних структур данных [Электронный ресурс]. – Режим доступа: https://studopedia.su/5\_13969\_razrabotka-vnutrennih-struktur-dannih.html. – Дата доступа: 30.05.2021.
3. Диаграмма вариантов использования (use case diagram) [Электронный ресурс]. – Режим доступа: http://khpi-iip.mipk.kharkiv.edu/library/case/leon/gl4/gl4.html. – Дата доступа: 30.05.2021.
4. Онлайн-сервис для управления логистикой, мобильным персоналом и выездными сервисами [Электронный ресурс]. – Режим доступа: https://maxoptra.ru. – Дата доступа: 30.05.2021.
5. Программа для транспортных компаний и владельцев транспорта - Первоисточник: https://www.kiberlog.ru/ [Электронный ресурс]. – Режим доступа: https://www.kiberlog.ru. – Дата доступа: 30.05.2021.
6. Программа для транспортных компаний и владельцев транспорта [Электронный ресурс]. – Режим доступа: https://yacurier.com/features/. – Дата доступа: 30.05.2021.
7. Мегалогист [Электронный ресурс]. – Режим доступа: https://mega-logist.ru. – Дата доступа: 30.05.2021.
8. АвтоПеревозки [Электронный ресурс]. – Режим доступа: http://www.autosoft.ru/products/transportation/. – Дата доступа: 30.05.2021.
9. 1С TMS Логистика [Электронный ресурс]. – Режим доступа: https://itob.ru. – Дата доступа: 30.05.2021.
10. «Интегрированные среды разработки программного обеспечения, основные функции и компоненты [Электронный ресурс]. – Режим доступа: https://infourok.ru/integrirovannye-sredy-razrabotki-programmnogo-obespecheniya-osnovnye-funkcii-i-komponenty-4994927.html. – Дата доступа: 30.05.2021.
11. Microsoft Visual Studio [Электронный ресурс]. – Режим доступа: https://www.tadviser.ru/index.php/Продукт:Microsoft\_Visual\_Studio. – Дата доступа: 30.05.2021.
12. Краткий обзор языка C# [Электронный ресурс]. – Режим доступа: https://docs.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp/. – Дата доступа: 30.05.2021.
13. Базы данных и модели данных. Основы проектирования реляционных баз данных [Электронный ресурс]. – Режим доступа: https://edu.tltsu.ru/sites/sites\_content/site216/html/media67139/theor\_bd.pdf. – Дата доступа: 30.05.2021.
14. Инфологическая модель [Электронный ресурс]. – Режим доступа: https://mobile.studbooks.net/2214795/informatika/infologicheskaya\_model. – Дата доступа: 30.05.2021.
15. Физическое проектирование базы данных [Электронный ресурс]. – Режим доступа: https://studbooks.net/2214796/informatika/fizicheskoe\_proektirovanie\_bazy\_dannyh. – Дата доступа: 30.05.2021.
16. Windows Forms [Электронный ресурс]. – Режим доступа: https://microsoft.fandom.com/ru/wiki/Windows\_Forms. – Дата доступа: 30.05.2021.
17. Паттерн MVC [Электронный ресурс]. – Режим доступа: https://professorweb.ru/my/WPF/documents\_WPF/level36/36\_3.php. – Дата доступа: 30.05.2021.
18. Что такое тестирование программного обеспечения? [Электронный ресурс]. – Режим доступа: https://www.it-courses.by/sqa/. – Дата доступа: 30.05.2021.

# Приложение А

(обязательное)

Текст программы

<?xml version="1.0" encoding="utf-8"?>

<configuration>

<configSections>

</configSections>

<connectionStrings>

<add name="Software\_tool\_for\_a\_logistics\_company.Properties.Settings.logisticConnectionString" connectionString="server=localhost;user id=root;password=1234;persistsecurityinfo=True;database=logistic" providerName="MySql.Data.MySqlClient"/>

</connectionStrings>

<startup>

<supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5.2"/>

</startup>

</configuration>

using System;

using System.Windows.Forms;

namespace Software\_tool\_for\_a\_logistics\_company

{

public partial class Brand : Form

{

public Brand()

{

InitializeComponent();

}

private void toolStripButton1\_Click(object sender, EventArgs e)

{

if (MessageBox.Show("Все изменения в системе будут сохранены в базе данных.\r\n" +

"Продолжить сохранение?",

"Подтвердите сохранение",

MessageBoxButtons.YesNo) == DialogResult.Yes)

try

{

маркаTableAdapter.Update(logisticDataSet.марка);

MessageBox.Show("Изменения сохранены");

loadData();

}

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

}

private void loadData()

{

// TODO: данная строка кода позволяет загрузить данные в таблицу "logisticDataSet.марка". При необходимости она может быть перемещена или удалена.

this.маркаTableAdapter.Fill(this.logisticDataSet.марка);

}

private void Brand\_Load(object sender, EventArgs e)

{

loadData();

}

private void Brand\_FormClosing(object sender, FormClosingEventArgs e)

{

Main.main.Show();

}

}

}

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;

namespace Software\_tool\_for\_a\_logistics\_company

{

public partial class Driver : Form

{

public Driver()

{

InitializeComponent();

}

private void Driver\_FormClosing(object sender, FormClosingEventArgs e)

{

Main.main.Show();

}

private void Driver\_Load(object sender, EventArgs e)

{

// TODO: данная строка кода позволяет загрузить данные в таблицу "logisticDataSet.трейлер". При необходимости она может быть перемещена или удалена.

this.трейлерTableAdapter.Fill(this.logisticDataSet.трейлер);

// TODO: данная строка кода позволяет загрузить данные в таблицу "logisticDataSet.заказ". При необходимости она может быть перемещена или удалена.

this.заказTableAdapter.Fill(this.logisticDataSet.заказ);

// TODO: данная строка кода позволяет загрузить данные в таблицу "logisticDataSet.водитель". При необходимости она может быть перемещена или удалена.

this.водительTableAdapter.Fill(this.logisticDataSet.водитель);

}

private void dataGridView1\_DataError(object sender, DataGridViewDataErrorEventArgs e)

{

loadData();

}

private void toolStripButton1\_Click(object sender, EventArgs e)

{

if (MessageBox.Show("Все изменения в системе будут сохранены в базе данных.\r\n" +

"Продолжить сохранение?",

"Подтвердите сохранение",

MessageBoxButtons.YesNo) == DialogResult.Yes)

try

{

водительTableAdapter.Update(logisticDataSet.водитель);

MessageBox.Show("Изменения сохранены");

loadData();

}

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

}

private void loadData()

{

// TODO: данная строка кода позволяет загрузить данные в таблицу "logisticDataSet.водитель". При необходимости она может быть перемещена или удалена.

this.водительTableAdapter.Fill(this.logisticDataSet.водитель);

}

private void dataGridView2\_DataError(object sender, DataGridViewDataErrorEventArgs e)

{

}

}

}

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;

namespace Software\_tool\_for\_a\_logistics\_company

{

public partial class Firm : Form

{

public Firm()

{

InitializeComponent();

}

private void Firm\_Load(object sender, EventArgs e)

{

loadData();

}

private void Firm\_FormClosing(object sender, FormClosingEventArgs e)

{

Main.main.Show();

}

private void toolStripButton1\_Click(object sender, EventArgs e)

{

if (MessageBox.Show("Все изменения в системе будут сохранены в базе данных.\r\n" +

"Продолжить сохранение?",

"Подтвердите сохранение",

MessageBoxButtons.YesNo) == DialogResult.Yes)

try

{

фирмаTableAdapter.Update(logisticDataSet.фирма);

MessageBox.Show("Изменения сохранены");

loadData();

}

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

}

private void loadData()

{

// TODO: данная строка кода позволяет загрузить данные в таблицу "logisticDataSet.фирма". При необходимости она может быть перемещена или удалена.

this.фирмаTableAdapter.Fill(this.logisticDataSet.фирма);

}

}

}

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;

namespace Software\_tool\_for\_a\_logistics\_company

{

public partial class Main : Form

{

public static Main main;

public Main()

{

InitializeComponent();

main = this;

}

private void Main\_Load(object sender, EventArgs e)

{

}

private void справочникМарокToolStripMenuItem\_Click(object sender, EventArgs e)

{

new Brand().Show();

this.Hide();

}

private void справочникФирмToolStripMenuItem\_Click(object sender, EventArgs e)

{

new Firm().Show();

this.Hide();

}

private void button1\_Click(object sender, EventArgs e)

{

new Trailer().Show();

this.Hide();

}

private void button2\_Click(object sender, EventArgs e)

{

new Driver().Show();

this.Hide();

}

private void button3\_Click(object sender, EventArgs e)

{

new Order().Show();

this.Hide();

}

private void button4\_Click(object sender, EventArgs e)

{

new MaxOrderCountByTrailer().Show();

this.Hide();

}

private void button5\_Click(object sender, EventArgs e)

{

new MaxOrderCount().Show();

this.Hide();

}

private void button6\_Click(object sender, EventArgs e)

{

new SummaryMoney().Show();

this.Hide();

}

}

}

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;

namespace Software\_tool\_for\_a\_logistics\_company

{

public partial class MaxOrderCount : Form

{

public MaxOrderCount()

{

InitializeComponent();

}

private void MaxOrderCount\_Load(object sender, EventArgs e)

{

// TODO: данная строка кода позволяет загрузить данные в таблицу "logisticDataSet.водитель". При необходимости она может быть перемещена или удалена.

this.водительTableAdapter.Fill(this.logisticDataSet.водитель);

// TODO: данная строка кода позволяет загрузить данные в таблицу "logisticDataSet.DriverOrderCount". При необходимости она может быть перемещена или удалена.

this.driverOrderCountTableAdapter.FillByDriverWithMaxOrder(this.logisticDataSet.DriverOrderCount);

}

private void dataGridView1\_DataError(object sender, DataGridViewDataErrorEventArgs e)

{

}

private void MaxOrderCount\_FormClosing(object sender, FormClosingEventArgs e)

{

Main.main.Show();

}

}

}

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;

namespace Software\_tool\_for\_a\_logistics\_company

{

public partial class MaxOrderCountByTrailer : Form

{

public MaxOrderCountByTrailer()

{

InitializeComponent();

}

private void MaxOrderCount\_Load(object sender, EventArgs e)

{

// TODO: данная строка кода позволяет загрузить данные в таблицу "logisticDataSet.трейлер". При необходимости она может быть перемещена или удалена.

this.трейлерTableAdapter.Fill(this.logisticDataSet.трейлер);

// TODO: данная строка кода позволяет загрузить данные в таблицу "logisticDataSet.TrailerOrderCount". При необходимости она может быть перемещена или удалена.

this.trailerOrderCountTableAdapter.FillByTrailerWithMaxOrderCount(this.logisticDataSet.TrailerOrderCount);

}

private void dataGridView1\_DataError(object sender, DataGridViewDataErrorEventArgs e)

{

}

private void MaxOrderCount\_FormClosing(object sender, FormClosingEventArgs e)

{

Main.main.Show();

}

}

}

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;

namespace Software\_tool\_for\_a\_logistics\_company

{

public partial class Order : Form

{

public Order()

{

InitializeComponent();

}

private void Order\_FormClosing(object sender, FormClosingEventArgs e)

{

Main.main.Show();

}

private void Order\_Load(object sender, EventArgs e)

{

loadData();

}

private void toolStripButton1\_Click(object sender, EventArgs e)

{

if (MessageBox.Show("Все изменения в системе будут сохранены в базе данных.\r\n" +

"Продолжить сохранение?",

"Подтвердите сохранение",

MessageBoxButtons.YesNo) == DialogResult.Yes)

try

{

заказTableAdapter.Update(logisticDataSet.заказ);

MessageBox.Show("Изменения сохранены");

loadData();

}

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

}

private void loadData()

{

// TODO: данная строка кода позволяет загрузить данные в таблицу "logisticDataSet.водитель". При необходимости она может быть перемещена или удалена.

this.водительTableAdapter.Fill(this.logisticDataSet.водитель);

// TODO: данная строка кода позволяет загрузить данные в таблицу "logisticDataSet.трейлер". При необходимости она может быть перемещена или удалена.

this.трейлерTableAdapter.Fill(this.logisticDataSet.трейлер);

// TODO: данная строка кода позволяет загрузить данные в таблицу "logisticDataSet.заказ". При необходимости она может быть перемещена или удалена.

this.заказTableAdapter.Fill(this.logisticDataSet.заказ);

}

private void dataGridView1\_DataError(object sender, DataGridViewDataErrorEventArgs e)

{

}

private void button2\_Click(object sender, EventArgs e)

{

заказBindingSource.Filter = "";

}

private void button1\_Click(object sender, EventArgs e)

{

заказBindingSource.Filter = "[Расстояние\_км] <500";

}

}

}

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;

namespace Software\_tool\_for\_a\_logistics\_company

{

public partial class SummaryMoney : Form

{

public SummaryMoney()

{

InitializeComponent();

}

private void SummaryMoney\_Load(object sender, EventArgs e)

{

// TODO: данная строка кода позволяет загрузить данные в таблицу "logisticDataSet.SummaryMoney". При необходимости она может быть перемещена или удалена.

this.summaryMoneyTableAdapter.FillBySummaruMoney(this.logisticDataSet.SummaryMoney);

}

private void button1\_Click(object sender, EventArgs e)

{

float sum = 0;

for (int i = 0; i < dataGridView1.Rows.Count; i++) {

try {

sum += float.Parse(dataGridView1[3, i].Value.ToString());

} catch(Exception ex) { }

}

MessageBox.Show($"Общая сумма по заказам составила {sum}");

}

private void SummaryMoney\_FormClosing(object sender, FormClosingEventArgs e)

{

Main.main.Show();

}

}

}

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;

namespace Software\_tool\_for\_a\_logistics\_company

{

public partial class Trailer : Form

{

public Trailer()

{

InitializeComponent();

}

private void Trailer\_FormClosing(object sender, FormClosingEventArgs e)

{

Main.main.Show();

}

private void Trailer\_Load(object sender, EventArgs e)

{

loadData();

}

private void dataGridView1\_DataError(object sender, DataGridViewDataErrorEventArgs e)

{

}

private void dataGridView2\_DataError(object sender, DataGridViewDataErrorEventArgs e)

{

}

private void toolStripButton1\_Click(object sender, EventArgs e)

{

if (MessageBox.Show("Все изменения в системе будут сохранены в базе данных.\r\n" +

"Продолжить сохранение?",

"Подтвердите сохранение",

MessageBoxButtons.YesNo) == DialogResult.Yes)

try

{

трейлерTableAdapter.Update(logisticDataSet.трейлер);

MessageBox.Show("Изменения сохранены");

loadData();

}

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

}

private void loadData()

{

// TODO: данная строка кода позволяет загрузить данные в таблицу "logisticDataSet.водитель". При необходимости она может быть перемещена или удалена.

this.водительTableAdapter.Fill(this.logisticDataSet.водитель);

// TODO: данная строка кода позволяет загрузить данные в таблицу "logisticDataSet.заказ". При необходимости она может быть перемещена или удалена.

this.заказTableAdapter.Fill(this.logisticDataSet.заказ);

// TODO: данная строка кода позволяет загрузить данные в таблицу "logisticDataSet.фирма". При необходимости она может быть перемещена или удалена.

this.фирмаTableAdapter.Fill(this.logisticDataSet.фирма);

// TODO: данная строка кода позволяет загрузить данные в таблицу "logisticDataSet.марка". При необходимости она может быть перемещена или удалена.

this.маркаTableAdapter.Fill(this.logisticDataSet.марка);

// TODO: данная строка кода позволяет загрузить данные в таблицу "logisticDataSet.трейлер". При необходимости она может быть перемещена или удалена.

this.трейлерTableAdapter.Fill(this.logisticDataSet.трейлер);

}

private void button2\_Click(object sender, EventArgs e)

{

loadData();

}

private void button1\_Click(object sender, EventArgs e)

{

this.трейлерTableAdapter.FillByTop5BySize(this.logisticDataSet.трейлер);

}

}

}

,

# Приложение Б

(обязательное)

Код создания базы данных

DROP DATABASE IF EXISTS `logistic`;

CREATE DATABASE IF NOT EXISTS `logistic`;

USE `logistic`;

DROP TABLE IF EXISTS `Водитель`;

CREATE TABLE `Водитель` (

`ID` INTEGER NOT NULL AUTO\_INCREMENT,

`ФИО` VARCHAR(255),

`Дата\_рождения` DATETIME,

`Номер паспорта` VARCHAR(255),

`Стаж работы` INTEGER DEFAULT 0,

`Классность` VARCHAR(255),

`Напарник` INTEGER,

PRIMARY KEY (`ID`),

FOREIGN KEY(`Напарник`) REFERENCES `Водитель`(`ID`)

) ENGINE=innodb DEFAULT CHARSET=utf8;

SET autocommit=1;

INSERT INTO `Водитель` (`ID`, `ФИО`, `Дата\_рождения`, `Номер паспорта`, `Стаж работы`, `Классность`) VALUES (1, 'Макарова Людмила Михайловна', '1972-12-27 00:00:00', '3128209704', 3, '1');

INSERT INTO `Водитель` (`ID`, `ФИО`, `Дата\_рождения`, `Номер паспорта`, `Стаж работы`, `Классность`) VALUES (2, 'Павловский Яков Владимирович', '1978-12-19 00:00:00', '5235157610', 2, '2');

INSERT INTO `Водитель` (`ID`, `ФИО`, `Дата\_рождения`, `Номер паспорта`, `Стаж работы`, `Классность`) VALUES (3, 'Федорова Полина Макаровна', '1972-12-20 00:00:00', '0986174027', 5, '3');

INSERT INTO `Водитель` (`ID`, `ФИО`, `Дата\_рождения`, `Номер паспорта`, `Стаж работы`, `Классность`) VALUES (4, 'Агеев Никита Всеволодович', '1987-02-05 00:00:00', '4545525920', 7, '4');

INSERT INTO `Водитель` (`ID`, `ФИО`, `Дата\_рождения`, `Номер паспорта`, `Стаж работы`, `Классность`) VALUES (5, 'Волков Матвей Викторович', '1998-05-22 00:00:00', '5630391472', 8, '3');

INSERT INTO `Водитель` (`ID`, `ФИО`, `Дата\_рождения`, `Номер паспорта`, `Стаж работы`, `Классность`) VALUES (6, 'Григорьев Александр Львович', '1970-03-10 00:00:00', '6943346060', 1, '1');

INSERT INTO `Водитель` (`ID`, `ФИО`, `Дата\_рождения`, `Номер паспорта`, `Стаж работы`, `Классность`) VALUES (7, 'Иванов Владимир Ильич', '1995-11-03 00:00:00', '0512115230', 10, '2');

INSERT INTO `Водитель` (`ID`, `ФИО`, `Дата\_рождения`, `Номер паспорта`, `Стаж работы`, `Классность`) VALUES (8, 'Воробьева София Дамировна', '1975-10-30 00:00:00', '6026264590', 6, '4');

INSERT INTO `Водитель` (`ID`, `ФИО`, `Дата\_рождения`, `Номер паспорта`, `Стаж работы`, `Классность`) VALUES (9, 'Макаров Арсений Матвеевич', '1997-12-17 00:00:00', '4478084155', 12, '2');

INSERT INTO `Водитель` (`ID`, `ФИО`, `Дата\_рождения`, `Номер паспорта`, `Стаж работы`, `Классность`) VALUES (10, 'Быкова Диана Александровна', '1990-01-09 00:00:00', '7888302398', 0, '3');

DROP TABLE IF EXISTS `Марка`;

CREATE TABLE `Марка` (

`Марка` VARCHAR(255) NOT NULL,

PRIMARY KEY (`Марка`)

) ENGINE=innodb DEFAULT CHARSET=utf8;

SET autocommit=1;

INSERT INTO `Марка` (`Марка`) VALUES ('Марка 1');

INSERT INTO `Марка` (`Марка`) VALUES ('Марка 2');

INSERT INTO `Марка` (`Марка`) VALUES ('Марка 3');

INSERT INTO `Марка` (`Марка`) VALUES ('Марка 4');

INSERT INTO `Марка` (`Марка`) VALUES ('Марка 5');

DROP TABLE IF EXISTS `Фирма`;

CREATE TABLE `Фирма` (

`Фирма` VARCHAR(255) NOT NULL,

PRIMARY KEY (`Фирма`)

) ENGINE=innodb DEFAULT CHARSET=utf8;

SET autocommit=1;

INSERT INTO `Фирма` (`Фирма`) VALUES ('Фирма 1');

INSERT INTO `Фирма` (`Фирма`) VALUES ('Фирма 2');

INSERT INTO `Фирма` (`Фирма`) VALUES ('Фирма 3');

INSERT INTO `Фирма` (`Фирма`) VALUES ('Фирма 4');

DROP TABLE IF EXISTS `Трейлер`;

CREATE TABLE `Трейлер` (

`ID` INTEGER NOT NULL AUTO\_INCREMENT,

`Номерной\_знак` VARCHAR(255),

`Марка` VARCHAR(255),

`Фирма` VARCHAR(255),

`Грузоподъемность` DOUBLE NULL DEFAULT 0,

`Расход\_на\_100\_км` DOUBLE NULL DEFAULT 0,

`Длина\_прицепа` DOUBLE NULL DEFAULT 0,

`Стоимость\_перевозки\_руб\_км` DECIMAL(19,4) DEFAULT 0,

PRIMARY KEY (`ID`),

FOREIGN KEY(`Марка`) REFERENCES `Марка`(`Марка`),

FOREIGN KEY(`Фирма`) REFERENCES `Фирма`(`Фирма`)

) ENGINE=innodb DEFAULT CHARSET=utf8;

SET autocommit=1;

INSERT INTO `Трейлер` (`ID`, `Номерной\_знак`, `Марка`, `Фирма`, `Грузоподъемность`, `Расход\_на\_100\_км`, `Длина\_прицепа`, `Стоимость\_перевозки\_руб\_км`) VALUES (1, 'С 005 Х У', 'Марка 1', 'Фирма 1', 800, 11.8, 6, 1.78);

INSERT INTO `Трейлер` (`ID`, `Номерной\_знак`, `Марка`, `Фирма`, `Грузоподъемность`, `Расход\_на\_100\_км`, `Длина\_прицепа`, `Стоимость\_перевозки\_руб\_км`) VALUES (2, 'С 303 Р С', 'Марка 4', 'Фирма 4', 200, 13.5, 8, 3);

INSERT INTO `Трейлер` (`ID`, `Номерной\_знак`, `Марка`, `Фирма`, `Грузоподъемность`, `Расход\_на\_100\_км`, `Длина\_прицепа`, `Стоимость\_перевозки\_руб\_км`) VALUES (3, 'А 200 У У', 'Марка 2', 'Фирма 2', 135, 9.6, 4, 5);

INSERT INTO `Трейлер` (`ID`, `Номерной\_знак`, `Марка`, `Фирма`, `Грузоподъемность`, `Расход\_на\_100\_км`, `Длина\_прицепа`, `Стоимость\_перевозки\_руб\_км`) VALUES (4, 'В 634 К Н', 'Марка 5', 'Фирма 4', 500, 15.6, 7, .45);

INSERT INTO `Трейлер` (`ID`, `Номерной\_знак`, `Марка`, `Фирма`, `Грузоподъемность`, `Расход\_на\_100\_км`, `Длина\_прицепа`, `Стоимость\_перевозки\_руб\_км`) VALUES (5, 'А 006 Х Н', 'Марка 2', 'Фирма 2', 680, 23.8, 5, 1.5);

INSERT INTO `Трейлер` (`ID`, `Номерной\_знак`, `Марка`, `Фирма`, `Грузоподъемность`, `Расход\_на\_100\_км`, `Длина\_прицепа`, `Стоимость\_перевозки\_руб\_км`) VALUES (6, 'С 783 Т Х', 'Марка 5', 'Фирма 3', 300, 18.4, 6, .58);

INSERT INTO `Трейлер` (`ID`, `Номерной\_знак`, `Марка`, `Фирма`, `Грузоподъемность`, `Расход\_на\_100\_км`, `Длина\_прицепа`, `Стоимость\_перевозки\_руб\_км`) VALUES (7, 'М 910 С М', 'Марка 2', 'Фирма 1', 20, 6.2, 2, .2);

INSERT INTO `Трейлер` (`ID`, `Номерной\_знак`, `Марка`, `Фирма`, `Грузоподъемность`, `Расход\_на\_100\_км`, `Длина\_прицепа`, `Стоимость\_перевозки\_руб\_км`) VALUES (8, 'Х 079 Х У', 'Марка 3', 'Фирма 4', 189, 7.7, 3, 2.2);

DROP TABLE IF EXISTS `Заказ`;

CREATE TABLE `Заказ` (

`ID` INTEGER NOT NULL AUTO\_INCREMENT,

`Пункт\_отправления` VARCHAR(255),

`Пункт\_назначения` VARCHAR(255),

`Расстояние\_км` DECIMAL(18,0) DEFAULT 0,

`Вес` DECIMAL(18,0) DEFAULT 0,

`Количество\_мест` INTEGER DEFAULT 0,

`Трейлер` INTEGER,

`Водитель` INTEGER,

`Дата\_создания` DATETIME,

`Дата\_выполнения` DATETIME,

PRIMARY KEY (`ID`),

FOREIGN KEY(`Трейлер`) REFERENCES `Трейлер`(`ID`),

FOREIGN KEY(`Водитель`) REFERENCES `Водитель`(`ID`)

) ENGINE=innodb DEFAULT CHARSET=utf8;

SET autocommit=1;

INSERT INTO `Заказ` (`ID`, `Пункт\_отправления`, `Пункт\_назначения`, `Расстояние\_км`, `Вес`, `Количество\_мест`, `Трейлер`, `Водитель`, `Дата\_создания`, `Дата\_выполнения`) VALUES (1, 'Пункт 1', 'Пункт 2', 450, 123, 1, 1, 1, '2021-01-25 00:00:00', '2021-01-25 00:00:00');

INSERT INTO `Заказ` (`ID`, `Пункт\_отправления`, `Пункт\_назначения`, `Расстояние\_км`, `Вес`, `Количество\_мест`, `Трейлер`, `Водитель`, `Дата\_создания`, `Дата\_выполнения`) VALUES (2, 'Пункт 3', 'Пункт 4', 840, 132, 2, 2, 9, '2021-02-02 00:00:00', '2021-04-15 00:00:00');

INSERT INTO `Заказ` (`ID`, `Пункт\_отправления`, `Пункт\_назначения`, `Расстояние\_км`, `Вес`, `Количество\_мест`, `Трейлер`, `Водитель`, `Дата\_создания`, `Дата\_выполнения`) VALUES (3, 'Пункт 5', 'Пункт 6', 720, 14, 2, 3, 2, '2021-03-15 00:00:00', '2021-03-15 00:00:00');

INSERT INTO `Заказ` (`ID`, `Пункт\_отправления`, `Пункт\_назначения`, `Расстояние\_км`, `Вес`, `Количество\_мест`, `Трейлер`, `Водитель`, `Дата\_создания`, `Дата\_выполнения`) VALUES (4, 'Пункт 7', 'Пункт 8', 340, 400, 1, 4, 5, '2021-03-26 00:00:00', '2021-03-29 00:00:00');

INSERT INTO `Заказ` (`ID`, `Пункт\_отправления`, `Пункт\_назначения`, `Расстояние\_км`, `Вес`, `Количество\_мест`, `Трейлер`, `Водитель`, `Дата\_создания`, `Дата\_выполнения`) VALUES (5, 'Пункт 9', 'Пункт 10', 10, 325, 1, 5, 3, '2021-04-22 00:00:00', '2021-04-29 00:00:00');

INSERT INTO `Заказ` (`ID`, `Пункт\_отправления`, `Пункт\_назначения`, `Расстояние\_км`, `Вес`, `Количество\_мест`, `Трейлер`, `Водитель`, `Дата\_создания`, `Дата\_выполнения`) VALUES (6, 'Пункт 11', 'Пункт 12', 499, 13, 2, 6, 6, '2021-04-28 00:00:00', '2021-04-28 00:00:00');

INSERT INTO `Заказ` (`ID`, `Пункт\_отправления`, `Пункт\_назначения`, `Расстояние\_км`, `Вес`, `Количество\_мест`, `Трейлер`, `Водитель`, `Дата\_создания`, `Дата\_выполнения`) VALUES (7, 'Пункт 13', 'Пункт 14', 501, 125, 2, 6, 3, '2021-05-11 00:00:00', '2021-05-15 00:00:00');

INSERT INTO `Заказ` (`ID`, `Пункт\_отправления`, `Пункт\_назначения`, `Расстояние\_км`, `Вес`, `Количество\_мест`, `Трейлер`, `Водитель`, `Дата\_создания`, `Дата\_выполнения`) VALUES (8, 'Пункт 15', 'Пункт 16', 1240, 123, 2, 8, 3, '2021-05-12 00:00:00', '2021-05-12 00:00:00');

INSERT INTO `Заказ` (`ID`, `Пункт\_отправления`, `Пункт\_назначения`, `Расстояние\_км`, `Вес`, `Количество\_мест`, `Трейлер`, `Водитель`, `Дата\_создания`, `Дата\_выполнения`) VALUES (9, 'Пункт 17', 'Пункт 18', 132, 135, 2, 4, 9, '2021-05-13 00:00:00', '2021-05-13 00:00:00');

INSERT INTO `Заказ` (`ID`, `Пункт\_отправления`, `Пункт\_назначения`, `Расстояние\_км`, `Вес`, `Количество\_мест`, `Трейлер`, `Водитель`, `Дата\_создания`, `Дата\_выполнения`) VALUES (10, 'Пункт 19', 'Пункт 20', 18, 156, 1, 3, 4, '2021-05-18 00:00:00', '2021-05-18 00:00:00');