Частное учреждение образования

«Колледж бизнеса и права»

ПРОГРАММА АВТОМАТИЗАЦИИ УЧЕТА ЗАЯВОК НА ГРУЗОПЕРЕВОЗКИ

ПОЯСнительная записка

к курсовому проекту по дисциплине

«Конструирование программ и языки программирования»

КП Т.992016.401

Руководитель проекта (Е. В. Багласова )

Учащийся (И. С. Миклашевич )

2022

Изм.

Лист

№ докум.

Подпись

Дата

Лист

2

КП Т.992016.401 ПЗ

Разраб.

Миклашевич И.С.

Провер.

Багласова Е.В.

.

.

*Разработка программного средства для учета заявок на грузоперевозки*

Лит.

Листов

КБП

124

у

[Введение 3](#_Toc106011257)

[1. Объектно-ориентированный анализ и проектирование системы 4](#_Toc106011258)

[1.1. Сущность задачи 4](#_Toc106011259)

[1.2. Проектирование модели 5](#_Toc106011260)

[2. Вычислительная система 8](#_Toc106011261)

[2.1 Требования к аппаратным и оперативным ресурсам 8](#_Toc106011262)

[2.2 Инструменты разработки 8](#_Toc106011263)

[3. Проектирование задачи 11](#_Toc106011264)

[3.1 Требования к приложению 11](#_Toc106011265)

[3.2 Концептуальный прототип 11](#_Toc106011266)

[3.3 Организация данных 15](#_Toc106011267)

[3.4 Функции: логическая и физическая организация 17](#_Toc106011268)

[3.5 Проектирование справочной системы приложения 19](#_Toc106011269)

[4 Описание программного средства 20](#_Toc106011270)

[4.1 Общие сведения 20](#_Toc106011271)

[4.2 Функциональное назначение 20](#_Toc106011272)

[4.3 Входные и выходные данные 20](#_Toc106011273)

[5 Методика испытаний 21](#_Toc106011274)

[5.1 Технические требования 21](#_Toc106011275)

[5.2 Функциональное тестирование 21](#_Toc106011276)

[6 Применение 30](#_Toc106011277)

[6.1 Назначение программы 30](#_Toc106011278)

[6.2 Условие применения 30](#_Toc106011279)

[6.3 Справочная система 30](#_Toc106011280)

[Заключение 32](#_Toc106011281)

[Список информационных источников 33](#_Toc106011282)

[Приложение А (обязательное) Текст программы 34](#_Toc106011283)

[Приложение Б (Справочное) Выходные документы 84](#_Toc106011284)

# Введение

Актуальность проблемы. На современном этапе развития рыночных отношений возникает потребность ускоренного развития производственной инфраструктуры (в том числе в транспортной сфере), обеспечивающей надежное обращение материальных ресурсов. При грузовых перевозках автомобильный транспорт участвует практически во всех взаимосвязях производителей и потребителей продукции производственного назначения и товаров народного потребления. Предоставлением услуг в сфере грузоперевозок занимаются автотранспортные предприятия. Каждое автотранспортное предприятие имеет в собственности либо аренде целый парк автомобилей различного назначения (бортовые грузовики, бензовозы, самосвалы и т. п.), обладающих различными грузоподъемностями и расходом топлива. На сегодняшний день активно разрабатываются корпоративные информационные системы, ориентированные на повышение эффективности управления транспортным предприятием. Автоматизированные системы широко используются для учета заявок на грузоперевозки, контроля проезда, выполнения диспетчерских и экспедиторских функций. Внедрение современных систем автоматизации учета транспортных средств - является одной из важных задач. Это позволит обеспечивать взаимосвязь различных операций в единой информационной системе. Спектр изучения вопросов автоматизации учета транспортных средств, достаточно широк и представляет, как теоретический, так и практический интерес.

Объектом исследования является учет заявок на грузоперевозки.

Предметом исследование является контроль и управление системой учета заявок на грузоперевозку.

Цель исследования разработать автоматизированное программное обеспечение на языке программирования C# по теме “Учет заявок на грузоперевозки”.

Задачи исследования:

* рассмотреть основы понятия «заявка на грузоперевозку»;
* изучить и проанализировать литературу по данной теме;
* разработать программное средство;
* описать созданное программное средство;
* разработать базу данных.
* разработать систему по учету заявок на грузоперевозки.

В первом разделе «Объектно-ориентированный анализ и проектирование системы» рассматривается сущность и актуальность поставленной задачи, описание существующих аналогов, проектирование модели, отображающей функциональную структуру объекта.

Второй раздел «Вычислительная система» имеет описание вычислительной системы, а именно: технические характеристики персонального компьютера, требования, которые будут предъявляться к персональному компьютеру, описание операционной системы, языка реализации и языка моделирования.

Третий раздел «Проектирование задачи» включает: требования к программному средству, концептуальный прототип, логическую и физическую структуры данных в контексте среды разработки, структуру и описание функций пользователя в рамках среды разрабатываемого программного средства, функции и элементы управления, проектирование справочной системы программного средства.

# Объектно-ориентированный анализ и проектирование системы

# Сущность задачи

Грузоперевозки - это процесс, вследствие которого совершается перемещение в какое-либо место ценных, хрупких, крупногабаритных, да и вообще, любых объектов с помощью какого-нибудь транспорта. В кругу перевозчиков, а также и заказчиках, под понятием стоимости, как правило, понимается оплата в одну сторону, какой-либо перевозки. Так же не стоить связывать понятие - грузоперевозки с арендой спецтехники или ее услугами, это разные сферы деятельности.

За все время существования человечества можно выделить три революции в развитии грузоперевозок:

То время, когда человек изобрел колесо. После этого, сам процесс грузоперевозки стал намного проще.

Период, в котором человек сумел приручить некоторых животных (появился домашний скот). Человеку не надо было тратить столько сил, сколько он тратил раньше на перевозку, изнуряя свой организм.

Естественно - изобретение и изготовление транспортных средств. Это позволило уменьшить время грузоперевозки в несколько раз и, опять-таки, уменьшило затраты человеческих сил. В настоящее время, когда технологии с каждым днем становятся все совершенней и совершенней, перевозить грузы можно в любых количествах и на любые расстояния.

Каждая грузоперевозка будет содержать такие пункты как:

* код перевозки;
* заказчик;
* поставщик;
* груз;
* транспорт;
* город;
* дата поставки;

Учет грузоперевозок будет оформлен в виде таблицы. Так же в табличном виде будет присутствовать информация об заказчике, поставщике, выбранном грузе, а также о выбранном транспортном средстве.

В программе будет реализовано разделение функционала приложения для сотрудника и администратора.

Администратору будет доступен полный функционал приложения, такой как, создание учета об грузоперевозке, используя данные уже существующие в базе данных. А также у него будет возможность вносить вправки в таблицы с данными, добавляя информацию или же удаляя ее. Еще стоит заметить, что у администратора будет возможность удаления или добавления новых сотрудников в базу данных.

У сотрудника функционал будет ограничен только созданием учета об грузоперевозке. Без возможности внесения изменения в базу данных.

Исходя из исследования предметной области, основными функциональными возможностями данного программного средства будут являться:

* регистрация в системе через учетную запись администратора, с указанием следующих данных: имя, фамилия, отчество, логин, пароль, должность и дата рождения;
* авторизация, позволяющая разделить функционал администратора и сотрудника;
* внесение изменений в таблицы с данными;
* создание учета об грузоперевозке;
* создание отчета, содержащего информацию о перевозке;

# Проектирование модели

Главной целью проектирования моделей является отображения функциональной структуры объекта, то есть производимые ими действия и связи между этими действиями. Наиболее распространенным средством моделирования данных являются диаграммы «Сущность-связь», которые предназначены для графического представления моделей данных разрабатываемой программной системы и предлагают некоторой набор стандартных обозначений для определения данных и отношения между ними. С помощью этого вида диаграмм можно описать отдельные компоненты концептуальной модели данных и совокупность взаимосвязей между ними, имеющих важное значение для разрабатываемой системы.

Основными понятиями данной нотации являются понятия сущности и связи. При этом под сущностью понимается произвольное множество реальных и абстрактных объектов, каждый из которых обладает одинаковыми свойствами и характеристиками. В этом случае каждый рассматриваемый объект может иметь уникальное имя или идентификатор, а также отличаться от других экземпляров данной сущности.

Связь определяется как отношение или некоторая ассоциация между отдельными сущностями. Примерами связей могут являться родственные отношения типа «отец-сын» или производственные отношения типа «начальник-подчиненный». Другой тип связей задается отношениями «иметь в собственности» или «обладать свойством».

Графическая модель строится таким образом, чтобы связи между отдельными сущностями отражали не только семантический характер соответствующего отношения, но и дополнительные аспекты обязательности связей, а также кратность участвующих в данных отношениях экземпляров сущностей.

Определим сущности для данного программного средства и построим диаграмму «Сущность-связь». Исследовав предметную область, можно выделить следующие сущности, относящиеся к данному проекту: «Клиенты», «Грузы», «Перевозка», «Водитель», «Транспорт», «Город».

Для сущности «Клиенты» атрибутами будут являться:

* название организации;
* телефон;
* электронная почта;
* улица;
* ИНН;
* номер заказа.

Для сущности «Грузы» атрибутами будут являться:

* код груза;
* наименование груза;
* вес;
* количество;
* цена за единицу.

Для сущности «Перевозка» атрибутами будут являться:

* код перевозки;
* клиент;
* груз;
* транспорт;
* город;
* дата поставки.

Для сущности «Водитель» атрибутами будут являться:

* ФИО.

Для сущности «Транспорт» атрибутами будут являться:

* код транспорта;
* номер;
* грузоперевозки;
* водитель.

Для сущности «Город» атрибутами будут являться:

* наименование.

Диаграмма «Сущность-связь» представлена на рисунке 1.1.

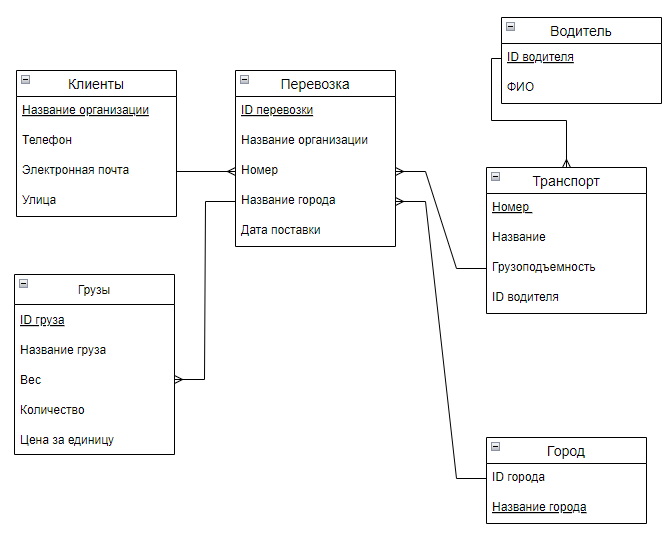


Рисунок 1.1 - Диаграмма «Сущность-связь» в нотации Баркера

Рисунок 1.1 – Диаграмма «Сущность-связь» в нотации Баркера

На основании проведенного анализа предметной области и выявленных функциональных возможностей, необходимо будет построить модель, которая бы отображала функциональную структуру объектов программного средства, действия, производимые ими и связь между этими действиями, а также определить инструменты разработки, которые будут использоваться в ходе написания программного средства.

В качестве инструмента для построения такой модели программного средства будет выбран унифицированный язык моделирования. Данный язык моделирования предназначен для специфицирования, визуализации, конструирования и документирования программных средств.

В рамках языка UML все представления о модели сложной системы фиксируются в виде специальных графических конструкций – диаграмм. В терминах языка UML определены следующие виды диаграмм: диаграмма вариантов использования, диаграмма классов, диаграмма деятельности, диаграмма последовательности, диаграмма компонентов.

Перечень этих диаграмм представляет собой неотъемлемую часть графической нотации языка UML, сам процесс объектно-ориентированного программирования (ООП) неразрывно связан с процессом построения этих диаграмм.

Суть диаграммы вариантов использования состоит в следующем: проектируемая система представляется в виде множества сущностей или актеров, взаимодействующих с системой с помощью вариантов использования.

Варианты использования описывают как взаимодействие между пользователем и сущностью, так и реакции сущности на получение отдельных сообщений от пользователей и восприятие этих сообщений за пределами сущности. Варианты использования могут включать в себя описание особенностей способов реализации сервиса и различных исключительных ситуаций, таких как корректная обработка ошибок системы. Множество вариантов использования в целом должно определять все возможные стороны ожидаемого поведения системы.

В данной проектируемой системе в качестве актера будет выступать сотрудник или администратор, который будет служить источником воздействия на моделируемую систему.

Диаграмма классов служит для предоставления статической структуры модели системы в терминологии классов объектно-ориентированного программирования. Диаграмма классов может отражать различные взаимосвязи между отдельными сущностями предметной области, такими как объекты и подсистемы, а также описывает их внутреннюю структуру и типы отношений. На данной диаграмме не указывается информация о временных аспектах функционирования системы.

Диаграмма классов представляет собой некоторой граф, вершинами которого являются элементы типа «классификатор», которые связаны различными типами структурных отношений. Когда говорят о данной диаграмме, имеют в виду статическую структурную модель проектируемой системы. Поэтому диаграмму классов принято считать графическим представлением таких структурных взаимосвязей логической модели системы, которые не зависят или инвариантны от времени.

Временный аспект поведения имеет существенное значение при моделировании синхронных процессов, описывающих взаимодействие объектов. Именно для этой цели и используются диаграммы последовательности, в которых ключевым моментом является динамика взаимодействия объектов во времени. При этом диаграмма последовательности имеет как бы два измерения: одно – слева направо в виде вертикальных линий, каждая из которых изображает линию жизни отдельного объекта, участвующего во взаимодействии; второе – вертикальная временная ось, направленная сверху вниз, на которой начальному моменту времени соответствует самая верхняя часть диаграммы.

В диаграмме последовательности для функции «Авторизация» актером выступает сотрудник или администратор, а сущностями, которые с ним взаимодействуют являются окно авторизации и база данных.

При моделировании поведения проектируемой или анализируемой системы возникает необходимость детализировать особенности алгоритмической и логической реализации выполняемых системой операций. Для моделирования процесса выполнения операций в языке UML используются диаграммы деятельности. Каждое состояние на диаграмме деятельности соответствует выполнению некоторой элементарной операции, переход в следующее состояние срабатывает только при завершении этой операции. Графически диаграмма деятельности представляется в форме графа, вершинами которого являются состояния, действия, а дугами – переходы от одного состояния действия к другому.

Основная цель использования диаграмм деятельности – визуализация особенностей реализации операций классов, когда необходимо представить алгоритмы их выполнения.

Диаграмма компонентов описывает объекты реального мира – компоненты программного обеспечения. Эта диаграмма позволяет определить архитектуру разрабатываемой системы, установив зависимости между программными компонентами:

* файл базы данных «AuthorizationBD.mdf»;
* файл базы данных «Gryzoperevozki.mdf»;
* файлы классов «ExcelHelper.cs»;
* файл приложения «YchetGryzoperevozok.exe»;
* файл справочной системы «help.chm»;
* файлы документов «DogovorNaDostavky.xls» и «PytevoiList.xls»;

# Вычислительная система

# Требования к аппаратным и оперативным ресурсам

Технические характеристики персонального компьютера, на котором будет вестись разработка:

* процессор Intel Core i5-10210U @ 1.60ГГц 2,11 ГГц;
* оперативная память 8 ГБ (DDR4);
* интегрированная видеокарта Intel(R) UHD Graphics;
* SSD диск на 256 ГБ;

Компьютер должен работать под управлением операционной системы, начиная с Windows 7 и новее. Для проведения различного рода испытаний наиболее удобной является Windows 10, так как она ориентируется, прежде всего, на максимальное использование возможностей персонального компьютера и обеспечение комфортных условий работы.

# Инструменты разработки

Ниже представлены инструменты разработки, которые будут использоваться в ходе написания программного средства.

Программное обеспечение:

* операционная система: Windows 10;
* среда разработки Microsoft Visual Studio 2022;
* язык программирования C#;
* система управления базами данных MS SQL Server;
* язык программирования SQL;
* модель проектирования пользовательского интерфейса Windows Form.

Операционная система – это набор управляющих программ, предназначенных для управления ресурсами вычислительной системы как единого комплекса. Иначе говоря, операционная система – набор программного обеспечения, который и обеспечивает работу компьютера, а также организует взаимодействие с пользователем. Основные функции операционной системы:

* управление файловой системой (запись, изменение, копирование файлов, контроль доступа);
* управление выполнением программ (распределение процессорного времени, загрузка программ с диска в оперативную память, перехват потенциально опасных действий);
* управление памятью (кэширование, распределение, контроль сохранности данных);
* диалог с пользователем (чтение команд с клавиатуры, с мыши, вывод информации на экран).

При разработке данного программного средства использовалась операционная система Windows 10, так как на данный момент она является самой распространенной операционной системой. В Windows 10 были исправлены практически все недостатки предыдущих операционных систем. Аппаратные требования Windows 10 скромнее, она способна работать даже на маломощных компьютерах, также добавлено множество функций, существенно облегчающих работу за компьютером. Причина заключается в хорошей оптимизации системы. В Windows 10 новые функциональные возможности добавляются с обновлениями компонентов, которые выходят два раза в год. Сначала организации используют эти выпуски обновлений компонентов для пилотного развертывания, чтобы обеспечить совместимость с существующими приложениями и инфраструктурой. Также Windows 10 намного лучше защищена, нежели ее предшественники, что является огромным плюсом.

Интегрированная среда разработки Visual Studio – это стартовая площадка для написания, отладки и сборки кода, а также последующей публикации приложений. Интегрированная среда разработки (IDE) представляет собой многофункциональную программу, которую можно использовать для различных аспектов разработки программного обеспечения. Помимо стандартного редактора и отладчика, которые существуют в большинстве сред IDE, Visual Studio включает в себя компиляторы, средства автозавершения кода, графические конструкторы и многие другие функции для упрощения процесса разработки.

Язык программирования C# – один из самых мощных, быстро развивающихся и востребованных языков в IT-отрасли. Является объектно-ориентированным языком, но поддерживает также и компонентно-ориентированное программирование. Разработка современных приложений все больше тяготеет к созданию программных компонентов в форме автономных и самоописательных пакетов, реализующих отдельные функциональные возможности. Главная особенность таких компонентов в том, что они представляют собой модель программирования со свойствами, методами и событиями. У них есть атрибуты, предоставляющие декларативные сведения о компоненте. Они включают в себя собственную документацию. Некоторые функции языка C#, обеспечивающие 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.

В C# существует единая система типов. Все типы C#, включая типы-примитивы, такие как int и double, наследуют от одного корневого типа object. Благодаря этому C# подходит для создания и применения программных компонентов.

Microsoft SQL Server — система управления реляционными базами данных (РСУБД), разработанная корпорацией Microsoft. Основной используемый язык запросов — Transact-SQL, создан совместно Microsoft и Sybase. Transact-SQL является реализацией стандарта ANSI/ISO по структурированному языку запросов (SQL) с расширениями. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия; конкурирует с другими СУБД в этом сегменте рынка.

Windows Forms – это платформа пользовательского интерфейса для создания классических приложений Windows. Она обеспечивает один из самых эффективных способов создания классических приложений с помощью визуального конструктора в Visual Studio. Такие функции, как размещение визуальных элементов управления путем перетаскивания, упрощают создание классических приложений. В Windows Forms можно разрабатывать графически сложные приложения, которые просто развертывать, обновлять, и с которыми удобно работать как в автономном режиме, так и в сети. Приложения Windows Forms могут получать доступ к локальному оборудованию и файловой системе компьютера, на котором работает приложение.

В Windows Forms можно разрабатывать графически сложные приложения, которые просто развертывать, обновлять, и с которыми удобно работать как в автономном режиме, так и в сети. Приложения Windows Forms могут получать доступ к локальному оборудованию и файловой системе компьютера, на котором работает приложение.

Для хранения данных используются различные системы управления базами данных. Однако, чтобы осуществлять связь между базой данных и приложением на C# необходим посредник. И именно таким посредником является технология ADO.NET. ADO.NET предоставляет собой технологию работы с данными, которая основана на платформе .NET Framework. Эта технология представляет нам набор классов, через которые мы можем отправлять запросы к базам данных, устанавливать подключения, получать ответ от базы данных и производить ряд других операций.

Возможность сетевой поддержки в данном программном средстве отсутствует, так как оно использует сервер с базой данных, который хранится, непосредственно, на самом компьютере.

# Проектирование задачи

# Требования к приложению

Разрабатываемое программное средство должно иметь понятный и удобный в использовании интерфейс, чтобы взаимодействия между программой и пользователем было максимально упрощено. Для обучения пользователей необходимо разработать справочную систему, в которой должны быть раскрыты все аспекты работы с программой, возможные трудности, возникшие во время работы и пути их решения. Также, в программном средстве будут реализованы функции, связанные с ведением базы данных, авторизацией, оформлением заявок на перевозку и формированием документов по оформленным заявкам.

Кроме того, при разработке окон необходимо соблюдать определенные требования: окна не должны быть перегружены информацией, должны иметь стандартные элементы управления данными, должны быть понятны пользователю. При конструировании окон в необходимых случаях нужно предусмотреть возможность защиты данных, установить ограничения на корректный ввод данных. Примером защиты будет служить окно авторизации, где необходимо будет ввести логин и пароль.

# Концептуальный прототип

Концептуальный прототип состоит из описания внешнего пользовательского интерфейса – системы меню, диалоговых окон и элементов управления.

Основной интерфейс программного средства будут представлять окна (Windows Form). Все окна будут содержать стандартные пользовательские элементы управления.

В программе предполагается создание навигационных кнопок.

В приложении было разработано главное меню, обеспечивающее быстрый и удобный доступ к функциям приложения и структурирующее их в однородные группы.

После запуска приложения будет появляться окно «Авторизации», представленное на рисунке 3.1.

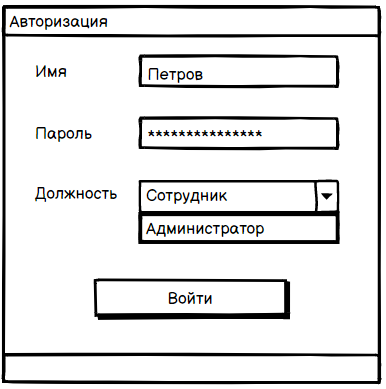


Рисунок 3.1 Прототип окна авторизации

Кнопка «Войти» будет открывать главное меню или панель администратора при определенном вводе информации в поле «Должность»

Прототип окна, служащего панелью администратора, «Админ панель» представлен на рисунке 3.2.

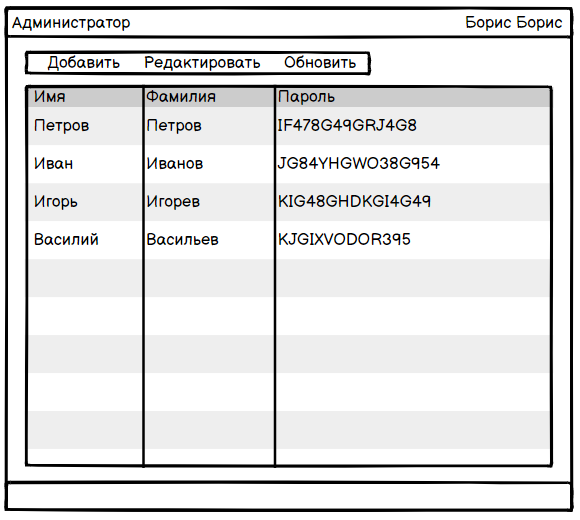


Рисунок 3.2 Прототип окна администратора

Кнопка «Добавить» будет добавлять информацию в таблицу.

Кнопка «Редактировать» будет позволять редактировать информацию в таблице.

Кнопка «Обновить» будет обновлять информацию в таблице.

Прототип окна, служащего главным меню, «Главное меню» представлен на рисунке 3.3.

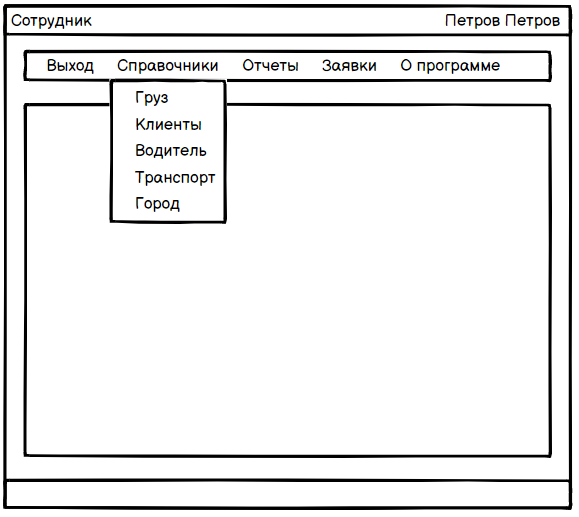


Рисунок 3.3 Прототип окна главного меню

Кнопка «Выход» будет закрывать программу.

Кнопка «Справочники» будет отображать меню, в котором можно будет выбрать один из справочников содержащий в себе информацию о данных хранящихся в БД.

Кнопка «Отчеты» будет открывать окно, содержащее в себе информацию об отчетах.

Кнопка «Заявки» будет открывать окно, содержащее в себе информацию об оформлении заявки на перевозку.

Кнопка «О программе» будет открывать окно, содержащее в себе информацию о программе.

Прототип окна одного из справочников системы, «Груз» представлен на рисунке 3.4.

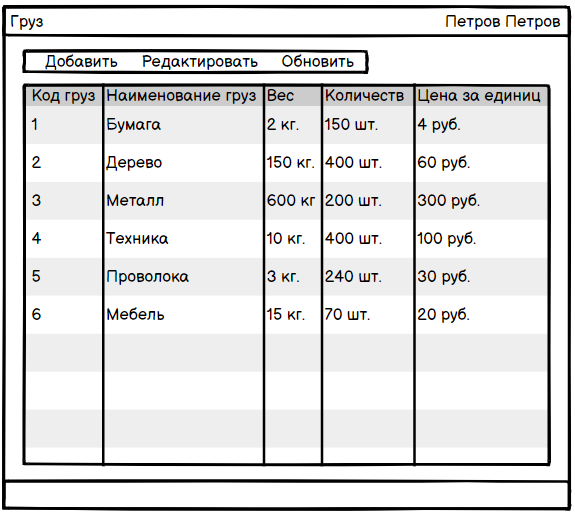


Рисунок 3.4 Прототип окна справочника

Кнопка «Добавить» будет добавлять информацию в таблицу.

Кнопка «Редактировать» будет позволять редактировать информацию в таблице.

Кнопка «Обновить» будет обновлять информацию в таблице.

Прототип окна, в котором оформляется заявка на перевозку, «Заявки» представлен на рисунке 3.5.

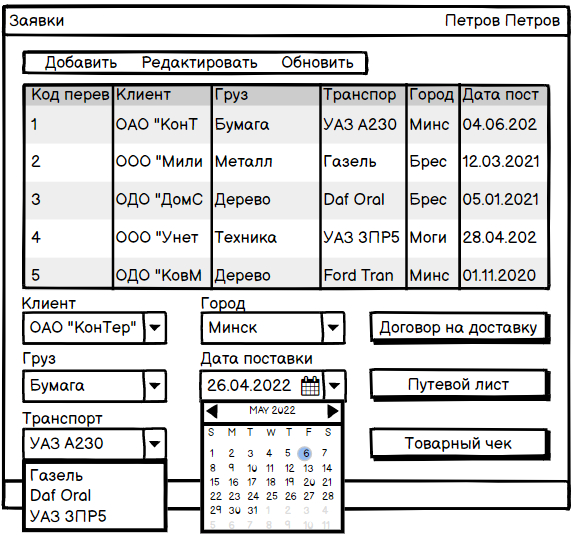


Рисунок 3.5 Прототип окна заявки

Кнопка «Добавить» будет добавлять информацию в таблицу.

Кнопка «Редактировать» будет позволять редактировать информацию в таблице.

Кнопка «Обновить» будет обновлять информацию в таблице.

Кнопка «Договор на доставку» будет создавать форму договора, на основе заложенного шаблона, используя информацию, введенную в таблице «Заявки».

Кнопка «Путевой лист» будет создавать форму договора, на основе заложенного шаблона, используя информацию, введенную в таблице «Заявки».

Кнопка «Товарный чек» будет создавать форму договора, на основе заложенного шаблона, используя информацию, введенную в таблице «Заявки».

# Организация данных

Реляционная модель основана на математическом понятии отношения, представлением которого является таблица. В реляционной модели отношения используются для хранения информации об объектах, представленных в базе данных. Отношение имеет вид двухмерной таблицы, в которой строки соответствуют записям, а столбцы – атрибутам. Каждая запись должна однозначно характеризоваться в таблице. Для этого используют первичные и вторичные ключи. Достоинством реляционной модели является простота и удобство физической реализации.

Реляционная модель базы данных подразумевает нормализацию всех таблиц данных. Нормализация – это формальный метод анализа отношений на основе их первичного ключа и функциональных зависимостей, существующих между атрибутами.

База данных соответствует реляционной модели данных, где каждый выделенной в ходе проектирования сущности соответствует таблица.

Структура базы данных разрабатываемого программного средства была создана с использованием СУБД встроенной в MS Visual Studio 2022 и включает 8 таблиц.

Таблица «PositionTable» хранит информацию о должностях. Структура приведена в таблице 3.1.

Таблица 3.1 – Структура таблицы «PositionTable»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер, байт | Описание |
| Id | int | 4 | Идентификатор должности |
| Position | nvarchar(50) | 100 | Название должности |

Таблица «EmployeesTable» хранит информацию о сотрудниках. Структура приведена в таблице 3.2.

Таблица 3.2 – Структура таблицы «EmployeesTable»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер, байт | Описание |
| Id | int | 4 | Идентификатор работника |
| Fio | nvarchar(50) | 100 | Имя, фамилия и отчество |
| Password | nvarchar(50) | 100 | Пароль |
| Position | nvarchar(50) | 100 | Название должности |

Таблица «CargoTable» хранит информацию о сотрудниках. Структура приведена в таблице 3.3.

Таблица 3.3 – Структура таблицы «CargoTable»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер, байт | Описание |
| Id cargo | int | 4 | Идентификатор груза |
| Name of cargo | nvarchar(50) | 100 | Название груза |
| Weight | int | 4 | Вес |
| Count | int | 4 | Количество |
| Price | int | 4 | Цена за единицу товара |
| Id transportation | int | 4 | Идентификатор заявки |

Таблица «CityTable» хранит информацию о сотрудниках. Структура приведена в таблице 3.4.

Таблица 3.4 – Структура таблицы «CityTable»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер, байт | Описание |
| Id | int | 4 | Идентификатор города |
| Name of city | nvarchar(50) | 100 | Название города |

Таблица «ClientTable» хранит информацию о сотрудниках. Структура приведена в таблице 3.5.

Таблица 3.5 – Структура таблицы «ClientTable»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер, байт | Описание |
| Id | int | 4 | Идентификатор клиента |
| Name of org | nvarchar(50) | 100 | Название организации |
| Telephone | nvarchar(50) | 100 | Телефон |
| Email | nvarchar(50) | 100 | Электронная почта |
| Street | nvarchar(50) | 100 | Улица |

Таблица «DriverTable» хранит информацию о сотрудниках. Структура приведена в таблице 3.6.

Таблица 3.6 – Структура таблицы «DriverTable»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер, байт | Описание |
| Id driver | int | 4 | Идентификатор водителя |
| Fio | nvarchar(50) | 100 | Фамилия, имя и отчество |

Таблица «TransportationTable» хранит информацию о сотрудниках. Структура приведена в таблице 3.7.

Таблица 3.7 – Структура таблицы «TransportationTable»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер, байт | Описание |
| Id transportation | int | 4 | Идентификатор заявки |
| Name of org | nvarchar(50) | 100 | Фамилия, имя и отчество |
| Nomer | nvarchar(50) | 100 | Номер автомобиля |
| Name of city | nvarchar(50) | 100 | Название города |
| Date | date | 3 | Дата поставки |

Таблица «TransportTable» хранит информацию о сотрудниках. Структура приведена в таблице 3.8.

Таблица 3.8 – Структура таблицы «TransportTable»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер, байт | Описание |
| Id | int | 4 | Идентификатор транспорта |
| Nomer | nvarchar(50) | 100 | Номер автомобиля |
| Name | nvarchar(50) | 100 | Название транспорта |
| Capacity | int | 4 | Грузоподъемность |
| Id driver | int | 4 | Идентификатор водителя |

# Функции: логическая и физическая организация

Для реализации графического интерфейса программного средства была создана единая структура Windows-окон, которые содержат все необходимые функции программного средства.

Обязательными функциями программного средства являются удаление, добавление и изменение данных.

Функции, которые выполняет программа – ведение данных в базе данных, поиск данных при помощи фильтра, формирование заказа.

Для успешного выполнения вышеперечисленных функций программное средство необходимо запустить на персональном компьютере (ПК). После запуска оно отобразит окно с авторизацией, где сотруднику потребуется выбрать свое ФИО и ввести пароль. В случае, если он не был зарегистрирован ранее, то это должен сделать администратор с целью создания учетной записи сотрудника. Затем, после успешной авторизации, перед ним откроется окно главного меню, где будет доступ ко всем функциям.

Данные необходимые для грузоперевозки выводятся в таблицах для удобства пользования графическим интерфейсом программного средства.

Функция «Добавить новую заявку» для последующего добавления заявки обрабатывается событием dgvTransportation\_CellContentClick, закрепленным за элементом управления DataGridView «Добавить». Данный компонент расположен в окне «Заявка на отправку». Код функции представлен ниже:

private void dgvTransportation\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

try

{

if (e.ColumnIndex == columnCountTransportation)

{

string task = dgvTransportation.Rows[e.RowIndex].Cells[columnCountTransportation].Value.ToString();

if (task == "Удалить")

{

if (MessageBox.Show("Удалить эту строку ?", "Удаление", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)

{

int rowIndex = e.RowIndex;

dgvTransportation.Rows.RemoveAt(rowIndex);

dataSetTransportation.Tables[tableTransportation].Rows[rowIndex].Delete();

sqlDataAdapterTransportation.Update(dataSetTransportation, tableTransportation);

}

}

else if (task == "Добавить")

{

int rowIndex = dgvTransportation.Rows.Count - 2;

DataRow row = dataSetTransportation.Tables[tableTransportation].NewRow();

row["Name of org"] = dgvTransportation.Rows[rowIndex].Cells["Name of org"].Value;

row["Nomer"] = dgvTransportation.Rows[rowIndex].Cells["Nomer"].Value;

row["Name of city"] = dgvTransportation.Rows[rowIndex].Cells["Name of city"].Value;

row["Date"] = dgvTransportation.Rows[rowIndex].Cells["Date"].Value;

dataSetTransportation.Tables[tableTransportation].Rows.Add(row);

dataSetTransportation.Tables[tableTransportation].Rows.RemoveAt(dataSetTransportation.Tables[tableTransportation].Rows.Count - 1);

dgvTransportation.Rows.RemoveAt(dgvTransportation.Rows.Count - 2);

dgvTransportation.Rows[e.RowIndex].Cells[columnCountTransportation].Value = "Удалить";

sqlDataAdapterTransportation.Update(dataSetTransportation, tableTransportation);

newRowAddingTransportation = false;

}

else if (task == "Обновить")

{

int r = e.RowIndex;

dataSetTransportation.Tables[tableTransportation].Rows[r]["Name of org"] = dgvTransportation.Rows[r].Cells["Name of org"].Value;

dataSetTransportation.Tables[tableTransportation].Rows[r]["Nomer"] = dgvTransportation.Rows[r].Cells["Nomer"].Value;

dataSetTransportation.Tables[tableTransportation].Rows[r]["Name of city"] = dgvTransportation.Rows[r].Cells["Name of city"].Value;

dataSetTransportation.Tables[tableTransportation].Rows[r]["Date"] = dgvTransportation.Rows[r].Cells["Date"].Value;

sqlDataAdapterTransportation.Update(dataSetTransportation, tableTransportation);

dgvTransportation.Rows[e.RowIndex].Cells[columnCountTransportation].Value = "Удалить";

}

ReloadDataTransportation();

}

}

catch (SqlException ex)

{

MessageBox.Show("Значение уже используется в другой таблице, поэтому не может быть удалено или изменено!", "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

Остальной код программы представлен в приложении А.

# Проектирование справочной системы приложения

Для работы с приложением начинающего пользователя необходимо обеспечить качественной справочной системой, в которой должны быть приведены методы и приемы работы с приложением, включающие данные о том, что произойдет после нажатия на определенную кнопку или при выборе определенного пункта меню; сведения о том, какую информацию и в каком виде следует вводить в соответствующие поля.

Справочная система необходима для ознакомления с программой. В ней должна присутствовать информация, которая может пригодиться пользователю: о возможностях приложения и о разработчике программы.

Система справки данного программного средства будет содержать следующие разделы:

* «1. О программе»;
* «2. Начало работы»;
* «3. Руководство администратора»;
* «4. Руководство сотрудника»;
* «5. О разработчике».

Справочная система будет создана с помощью программного средства HTML Help Workshop.

В разделе «1. О программе» предоставляется информация о программном средстве, а именно: предназначение, цели создания, описание возможностей.

В разделе «2. Начало работы» содержится информация о том, как запустить приложение, об окне, которое появится после запуска программного средства, а также описание элементов, находящихся в нем.

В разделе «3. Руководство администратора» предоставлена информация обо всех возможностях администратора, в том числе и описание всех кнопок, полей и окон, с которыми взаимодействует администратор.

В разделе «4. Руководство сотрудника» содержится информация, а также описание всех кнопок, окон и полей, которые доступны для взаимодействия сотруднику.

В разделе «5. О разработчике» предоставлена информация о разработчике.

# Описание программного средства

# Общие сведения

Разрабатываемое программное средство предназначено для учета заявок грузоперевозок, фильтрации и работы с информацией, касающейся непосредственно этих заявок грузоперевозок, заказов и сотрудников.

Программное средство позволяет добавлять, удалять и изменять данные, то есть, иначе говоря, осуществляет ведение базы данных.

Данное программное средство было разработано при использовании персонального компьютера со следующей конфигурацией:

* процессор Intel Core i5-10210U 1,6 ГГц;
* интегрированная видеокарта Intel(R) UHD Graphics;
* оперативная память 8 ГБ (DDR4);
* SSD диск на 256 ГБ;

Программное средство создано в среде разработки Visual Studio 2022 на языке программирования C# в операционной системе Windows 10. Программное средство может работать в средах операционных систем семейства Microsoft Windows, начиная с Windows 10. Программа не требовательна к системным ресурсам компьютера, также очень проста в использовании. Обладание специальными профессиональными навыками для работы с ней не требуется. Для работы данного программного средства необходима предварительная установка и настройка следующих программных продуктов:

* система управления базами данных Microsoft SQL Server 2019;
* платформа Microsoft .Net 6.0.

Примерный размер программного средства составляет 0 Мбайт

# Функциональное назначение

Программное средство разрабатывалось для сотрудников с целью упрощения оформление заявок на грузоперевозки, путём ведения базы данных, а также повышения их работоспособности.

Программа использует стандартные элементы управления, такие как кнопки, меню, поля ввода, таблицы, что обеспечивает единство интерфейса системы и программного средства.

В работе программного средства предусмотрены некоторые ситуации, которые должны предупреждать сотрудника, чтобы он выполнял все необходимые требования по эксплуатации программы. Для этого существуют сообщения системы. Например, если сотрудник заполнил не все поля при оформлении заказа, то на экран выведется сообщение, где будет указано об отсутствии информации в том или ином поле.

Таким образом, программа может применяться в реальных условиях, представляя собой достаточно удобный помощник.

# Входные и выходные данные

В данном программном средстве входными данными являются: информация о грузах, о клиентах, о городах о водителях и транспорте.

Входными данными при добавлении груза являются:

* название груза;
* вес;
* количество;
* цена за единицу;

Входными данными при добавлении клиента являются:

* название организации;
* телефон;
* электронная почта;
* улица;

Входными данными при добавлении города являются:

* название города;

Входными данными при добавлении водителя являются:

* фио;

Входными данными при добавлении транспорта являются:

* номер;
* название;
* грузоподъемность;
* id водителя;

Выходными данными являются путевой лист, который представлены в приложении Б на рисунке Б.1, а также договор на доставку, представленный на рисунках Б.2.

# Методика испытаний

# Технические требования

Минимальными требованиями для оптимальной работы программного средства является персональный компьютер (ПК) со следующими характеристиками:

* процессор Intel Core i5-10210U 1,6 ГГц;
* интегрированная видеокарта Intel(R) UHD Graphics;
* оперативная память 8 ГБ (DDR4);
* SSD диск на 256 ГБ;
* программа для работы с таблицами Microsoft Excel 2016.

# Функциональное тестирование

Данное тестирование проводится для выявления неполадок и недочетов программы на этапе ее сдачи в эксплуатацию.

Функциональное тестирование предполагает проверку выполнения определенных на этапе проектирования функций.

Начальное окно, оно же окно авторизации, представлено на рисунке 5.1.

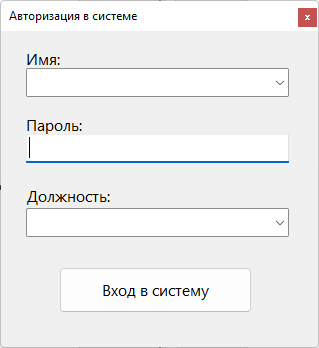


Рисунок 5.1 – Окно авторизации

Протестируем функцию авторизации. Для этого, находясь в данном окне, введем Имя и Пароль, после чего нажмем на кнопку «Войти в систему».

Таблица 5.1 – Тест-кейсы для функции «Авторизация»

|  |  |  |  |
| --- | --- | --- | --- |
| № | Модуль/функция | Шаги воспроизведения | Результат |
| 1 | Авторизация | 1. Заполнить поля данными:   * Имя: не заполняем * Пароль: «Qwerty1» * Должность: не заполняем   2. Нажать на кнопку «Войти в систему» | Ожидаемый результат: отображение предупреждения, представленного на рисунке 5.2. |
| Фактический результат: соответствует ожидаемому |
| 2 | Авторизация | 1. Заполнить поля данными:   * Имя: «Васильчук Николай Петрович» * Пароль: «1» * Должность: «Администратор»   2. Нажать на кнопку «Войти в систему» | Ожидаемый результат: отображение предупреждения, представленного на рисунке 5.3. |
| Фактический результат: соответствует ожидаемому |

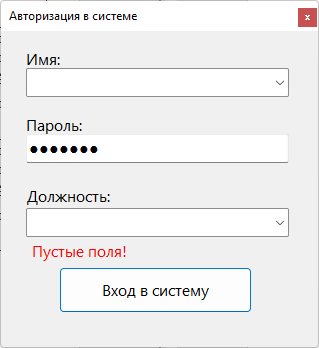


Рисунок 5.2 – Предупреждение, связанное с вводом неверных данных при авторизации

В случае если авторизация осуществлялась от имени администратора, то окно будет выглядеть также, как на рисунке 5.3. В случае если авторизация осуществлялась от имени обычного сотрудника, то откроется главное окно приложения.

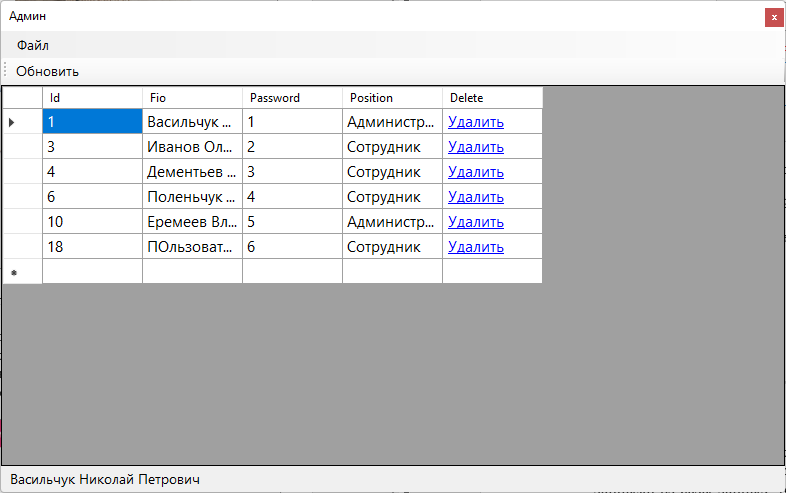


Рисунок 5.3 – Окно Администратора

Протестируем функцию изменения данных учетной записи. Для этого, находясь в окне Администратора, введем новые данные представленные на рисунке 5.4.

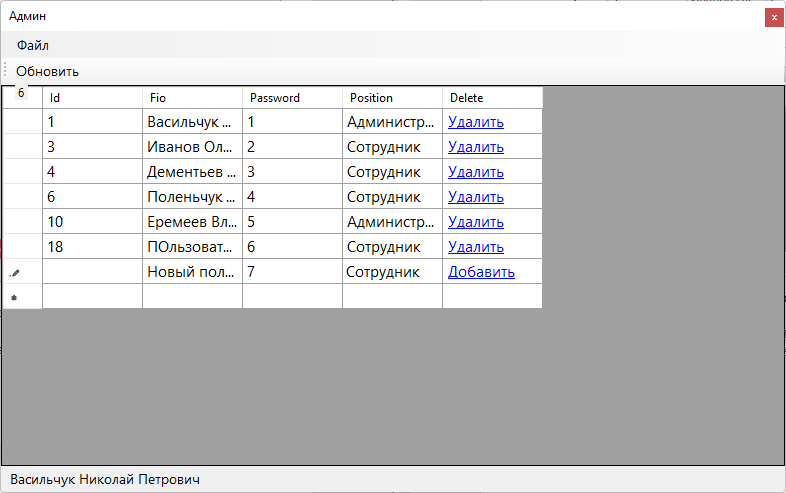


Рисунок 5.4 – Окно изменения данных учетной записи

Тест-кейсы для функции изменения данных учетной записи представлены в таблице 5.2.

Таблица 5.2 – Тест-кейсы для функции «Добавление данных учетной записи»

|  |  |  |  |
| --- | --- | --- | --- |
| № | Модуль/функция | Шаги воспроизведения | Результат |
| 3 | Изменение данных учетной записи | 1. Изменить данные полей:   * ФИО: «Новый пользователь» * Пароль: «7» * Должность: «Сотруд»   2. Нажать на кнопку «Добавить» | Ожидаемый результат: отображение предупреждения, представленного на рисунке 5.5. |
| Фактический результат: соответствует ожидаемому |
| 4 | Изменение данных учетной записи | 1. Изменить данные полей:   * ФИО: «Новый пользователь» * Пароль: «7» * Должность: «Сотрудник»   2. Нажать на кнопку «Добавить» | Ожидаемый результат: отображение предупреждения, представленного на рисунке 5.6. |
| Фактический результат: соответствует ожидаемому |

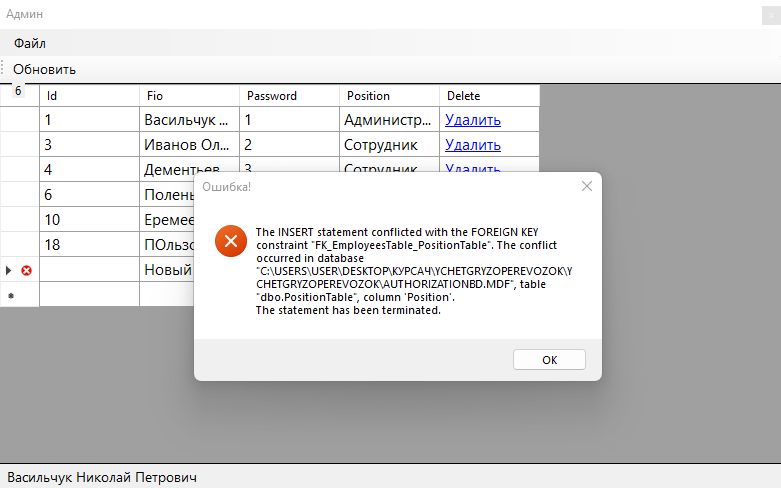


Рисунок 5.5 – Ошибка из-за ввода несуществующей должности.

В случае, если администратор успешно добавит данные, то результатом будет отображение нового пользователя в таблице, представленное на рисунке 5.6.

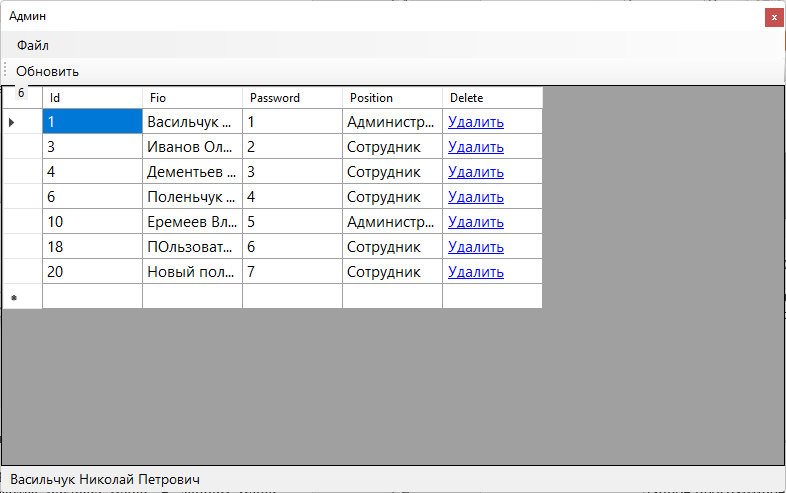


Рисунок 5.6 – Окно с успешным добавленным пользователем.

Протестируем функцию изменения данных сотрудников. Для этого, выделив необходимую строку таблицы, нажмем на кнопку «Обновить».

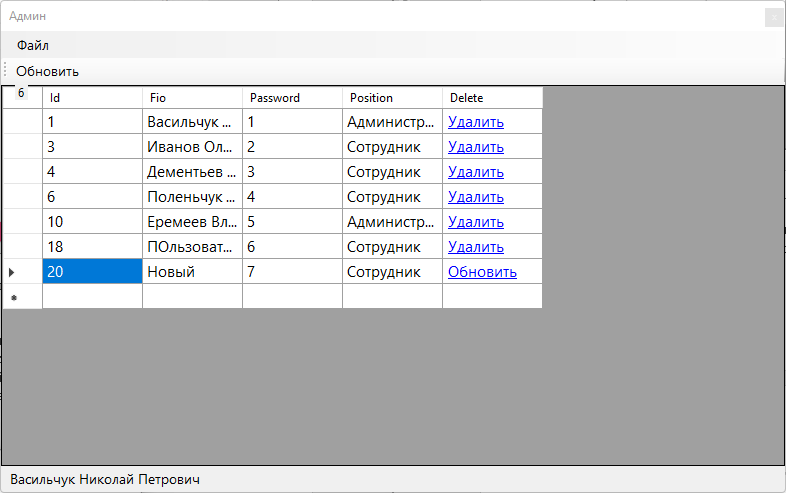


Рисунок 5.7 – Окно с измененными данными, но не нажатой кнопкой "Обновить"

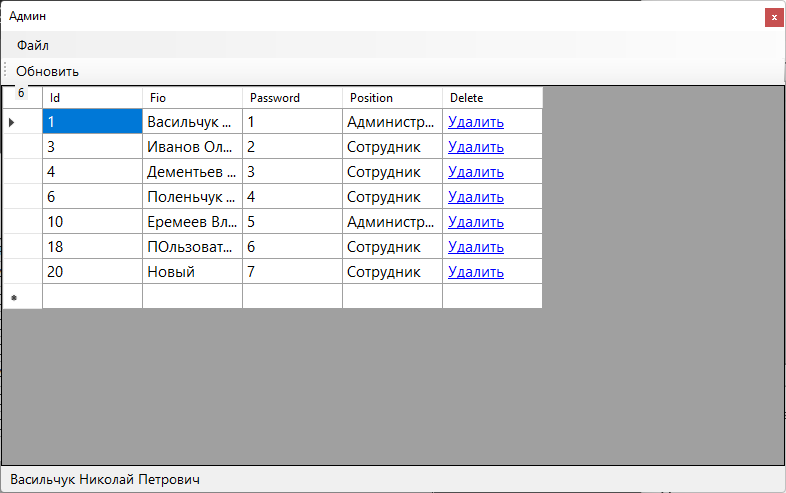


Рисунок 5.8 – Окно с обновленными данными.

Протестируем функцию удаления сотрудников. Выберем в списке необходимого для удаления сотрудника и нажмем на кнопку «Удалить». Результат представлен на рисунке 5.9.

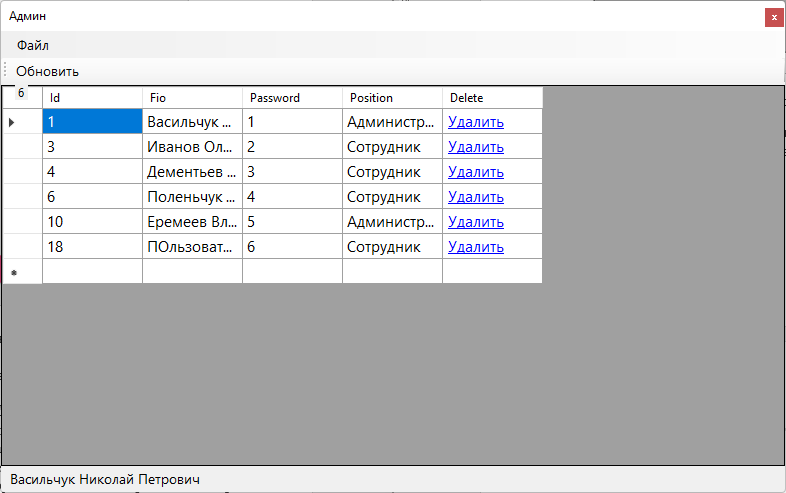


Рисунок 5.9 – Окно после удаления пользователя.

Протестируем функцию создания договора на поставку. Для этого нажмем на кнопку «Договор на поставку» в правом нижнем углу.

Тест-кейсы для функции создания документа на поставку представлены в таблице 5.3.

Таблица 5.3 – Тест-кейсы для функции создания документа на поставку.

|  |  |  |  |
| --- | --- | --- | --- |
| № | Модуль/функция | Шаги воспроизведения | Результат |
| 15 | Добавление договора на поствку | 1. Не заполняем данные полей окна.  3. Нажать на кнопку «Договор на поставку» | Ожидаемый результат: отображение предупреждения, представленного на рисунке 5.10. |
| Фактический результат: соответствует ожидаемому |
| 16 | Добавление договора на поствку | 1. Нажимаем на необходимую нам заявку.  3. Нажать на кнопку «Договор на поставку» | Ожидаемый результат: отображение предупреждения, представленного на рисунке 5.11. |
| Фактический результат: соответствует ожидаемому |

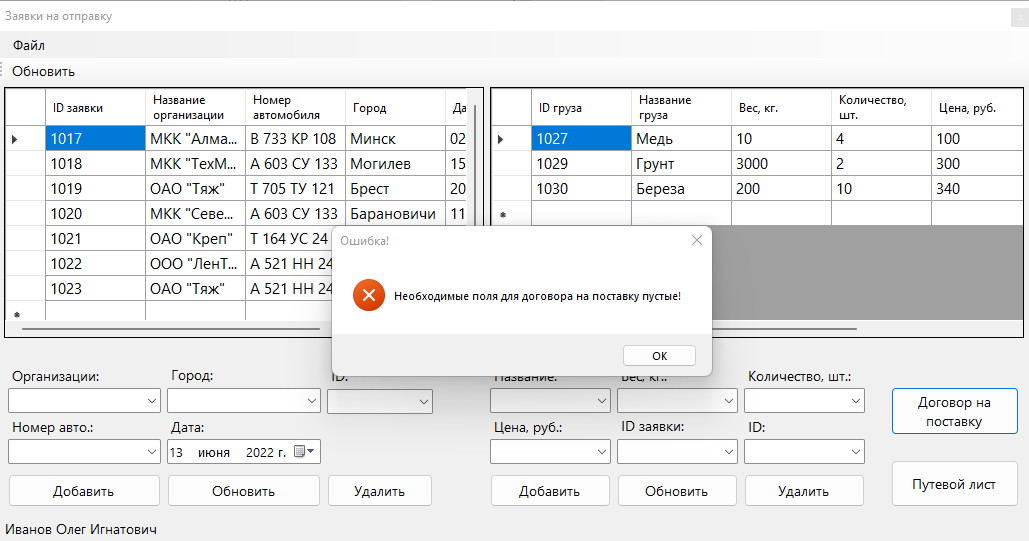


Рисунок 5.10 – Предупреждение о том, что единицы измерения не были выбраны

После успешного нажатия на кнопку «Договор на поставку» открывается проводник, представленный на рисунке 5.11.

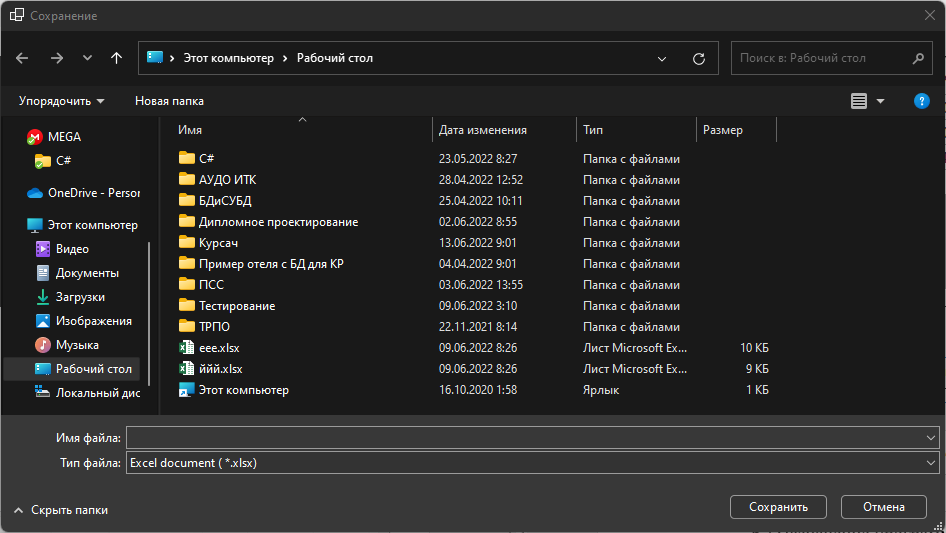


Рисунок 5.11 – Сохранение договора на поставку.

Выбираем имя файла, выбираем путь сохранения и нажимаем на кнопку «Сохранить»

Проверим функцию открытия EXCEL файла. Для этого нажмем на кнопку «Отчеты». Результат изображен на рисунке 5.12.

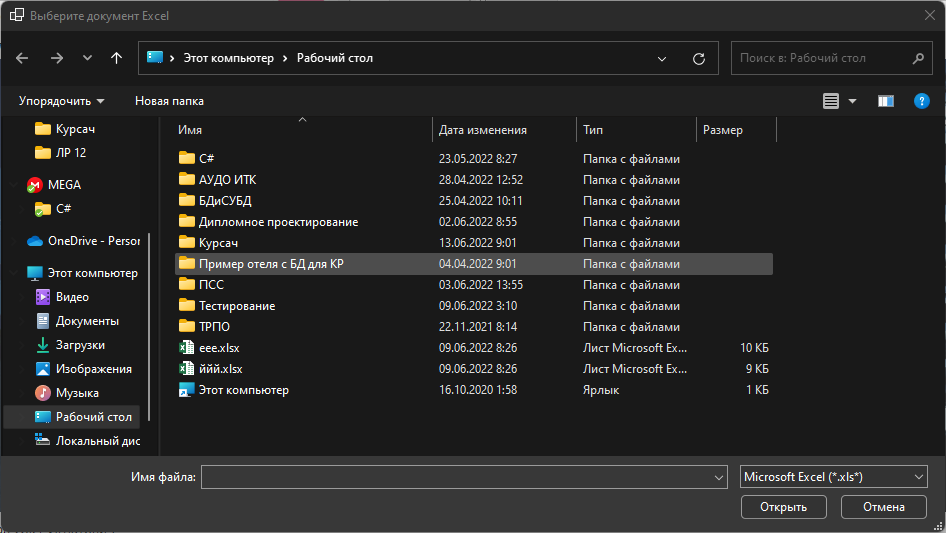


Рисунок 5.12 – Открытие EXCEL документа.

Выбираем файл “ййй.xlsx” и нажимаем на кнопку «Открыть». Откроется EXCEL файл. Результат изображен на рисунке 5.13.

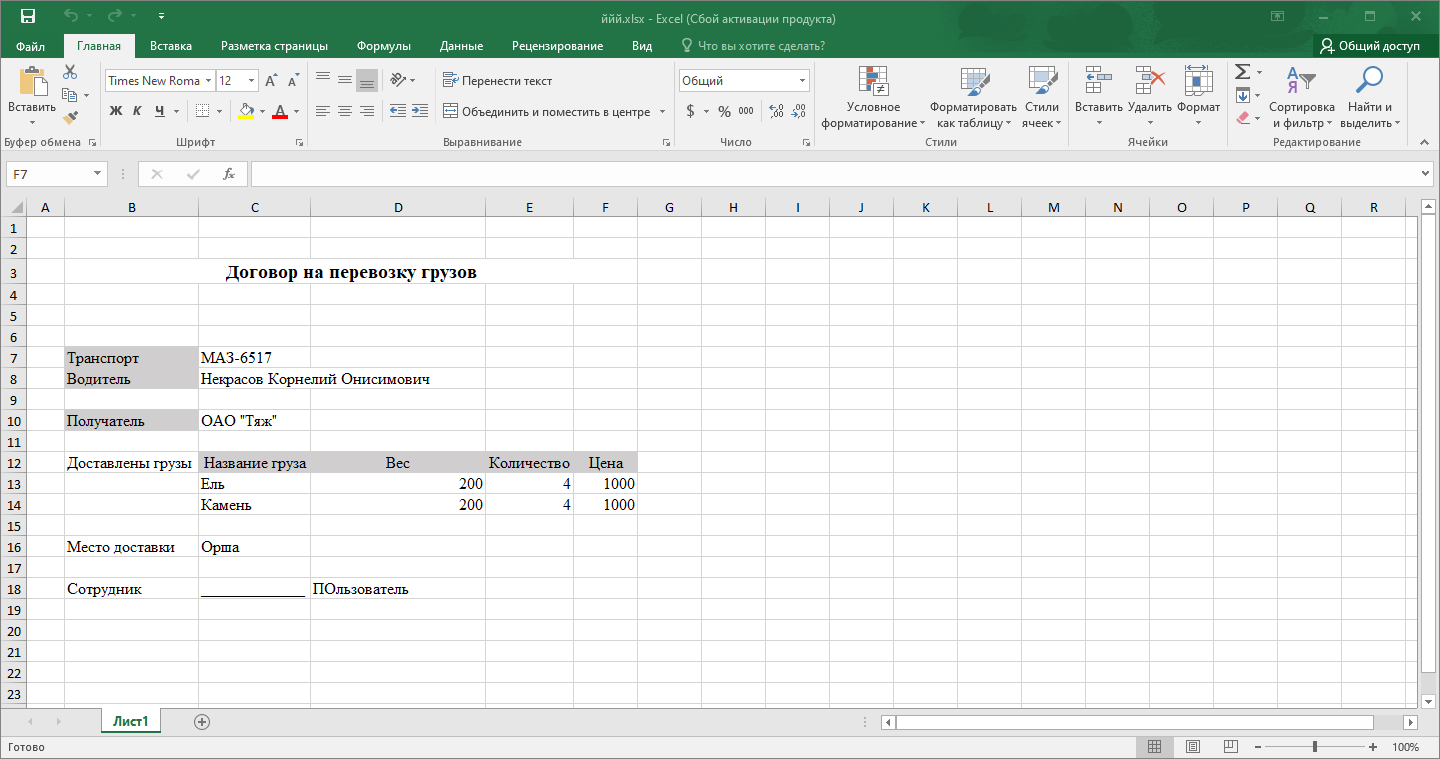


Рисунок 5.13 – Результат сохраненного документа на поставку

# Применение

# Назначение программы

Данное программное средство разработано для сотрудников условной компании по оформлению заявок на грузоперевозки.

Основной целью программного средства является обеспечение удобной работы с данными из базы данных, то есть осуществление фильтрации грузов, оформление заявок, а также с данными о сотрудниках, о клиентах, о транспорте и городах, а также создание документов на их основании. В том числе повышение работоспособности сотрудников, а также экономия времени на создание чеков по сформированным заявкам.

База данных проекта содержит достаточный объем необходимой информации для реализации поставленных задач.

Предоставлена возможность быстрого добавления новых заявок, клиентов, грузов, изменения данных в этих заявках, транспорте и водителях, оформление заявок, создание документов по данным заявок.

# Условие применения

Для бесперебойной работы данного программного средства необходимы следующие технические требования:

* процессор Intel Core i5-10210U 1,6 ГГц;
* интегрированная видеокарта Intel(R) UHD Graphics;
* оперативная память 8 ГБ (DDR4);
* SSD диск на 256 ГБ;
* операционная система Windows 10 и новее;
* платформа Microsoft .Net 6.0;
* клавиатура;
* мышь;
* программа для работы с таблицами Microsoft Excel 2016;
* система управления базами данных Microsoft SQL Server 2019.

# Справочная система

Справочная система программного средства представляет собой отдельный файл «help.chm» с полным описанием основных функций программы в формате \*.chm. В справочной системе даны ответы на вопросы, которые могут возникнуть при работе с приложением, что, несомненно, должно помочь при освоении программного средства.

Справочная система содержит следующие разделы:

* «1. О программе»;
* «2. Начало работы»;
* «3. Руководство администратора»;
* «4. Руководство сотрудника»;
* «5. О разработчике».

В разделе «1. О программе» представляется информация о программном средстве, а именно: его предназначение, цели его создания, описание его возможностей. Данный раздел представлен на рисунке 6.1.

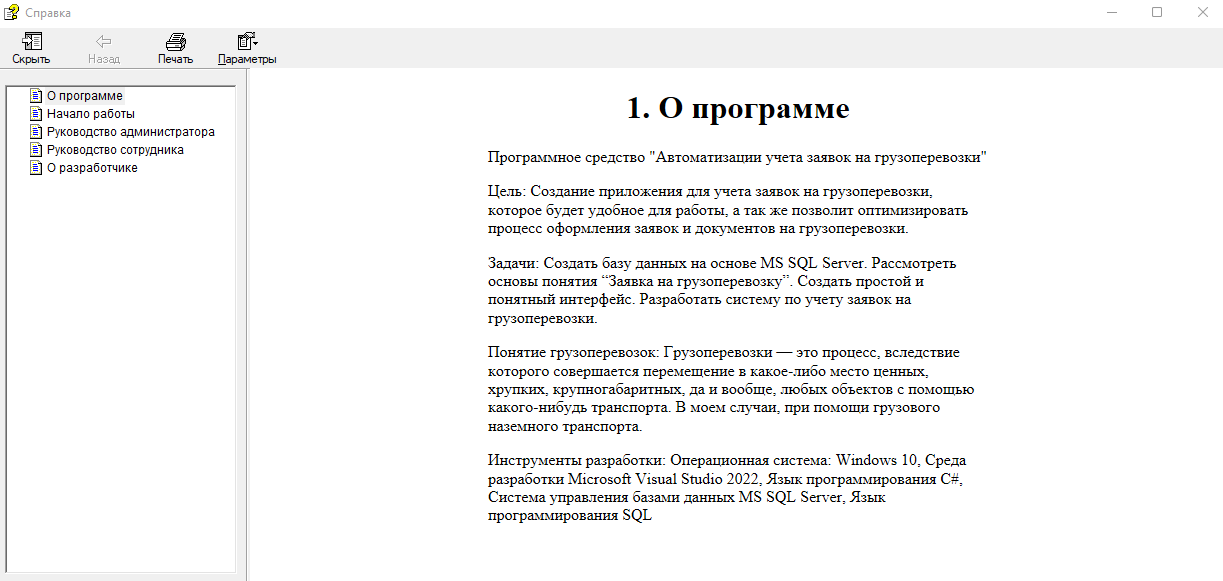


Рисунок 6.1 – О программе

В разделе «2. Начало работы» содержится информация о том, как запустить приложение, об окне, которое появится после запуска программного средства, а также описание элементов, находящихся в нем. Данный раздел представлен на рисунке 6.2.

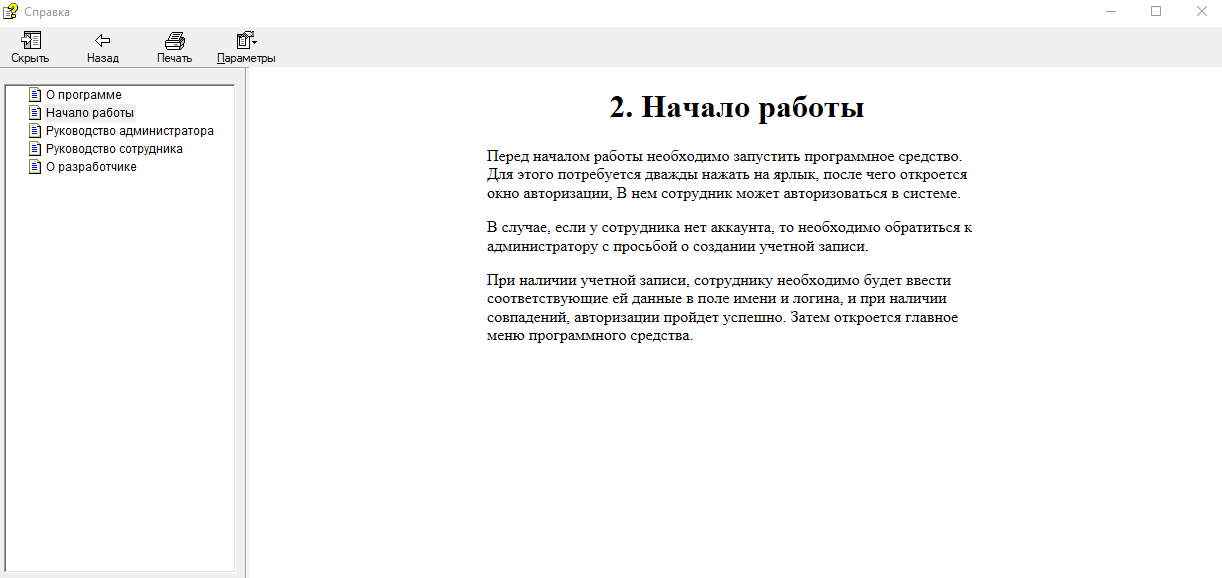


Рисунок 6.2 – Начало работы

В разделе «3. Руководство администратора» представлена информация обо всех возможностях администратора, в том числе и описание всех кнопок, полей и окон, с которыми взаимодействует администратор. Данный раздел представлен на рисунке 6.3.

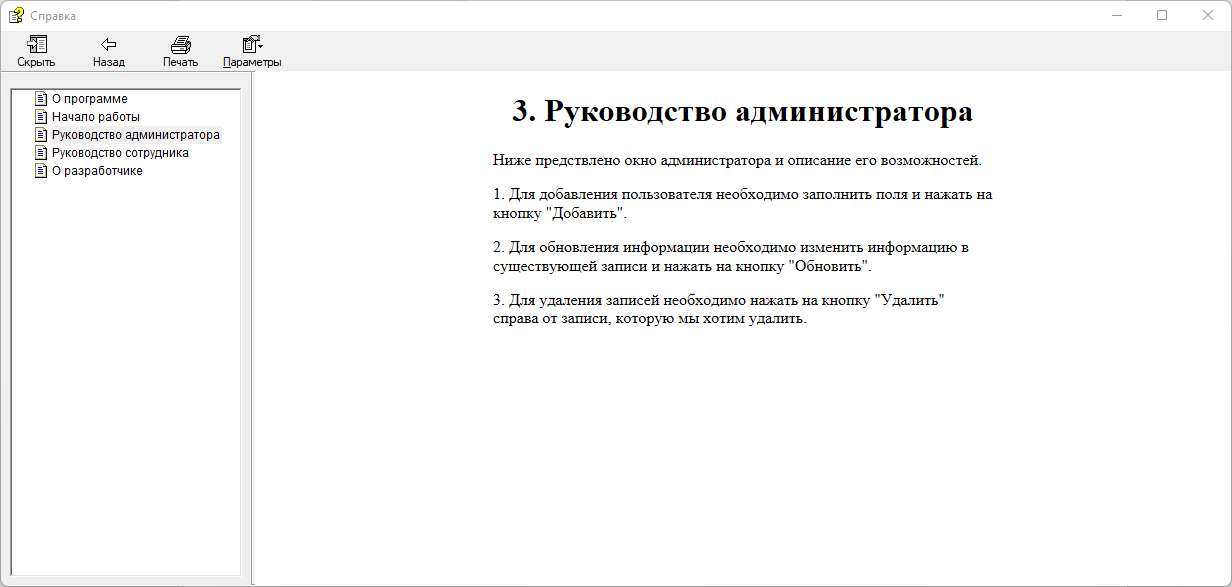


Рисунок 6.3 – Руководство администратора

В разделе «4. Руководство сотрудника» содержится информация, а также описание всех кнопок, окон и полей, которые доступны для взаимодействия сотруднику. Данный раздел представлен на рисунке 6.4.

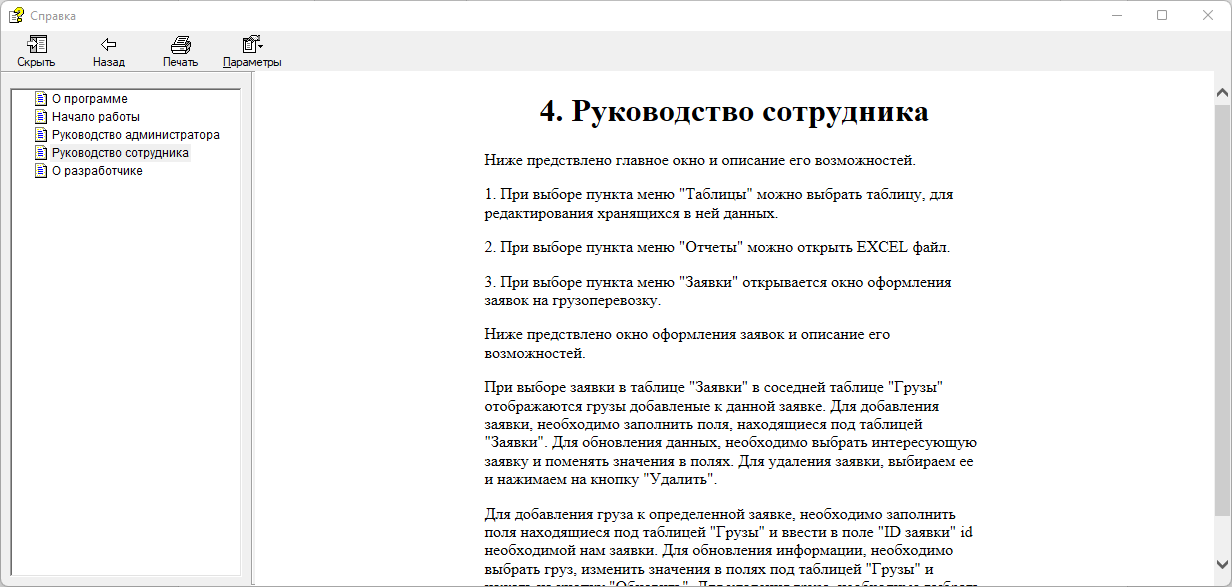


Рисунок 6.4 – Руководство сотрудника

В разделе «5. О разработчике» представлена информация о разработчике. Данный раздел представлен на рисунке 6.5.

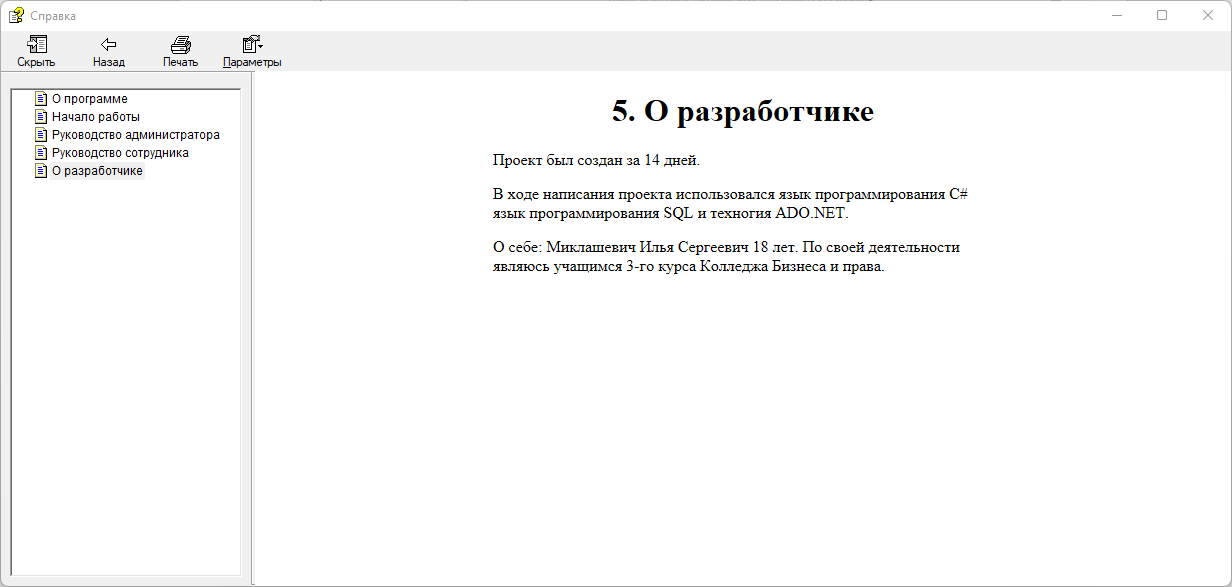


Рисунок 6.5 – О разработчике

# Заключение

В рамках курсового проектирования была произведена работа над проектом «Разработка программного средства для учета заявок на грузоперевозки».

Для достижения цели курсового проектирования были решены следующие задачи:

* определена вычислительная система, необходимая для создания программного средства;
* разработана физическая и логическая модель данных;
* по модели выполнено проектирование задачи;
* разработано программное средство;
* описано созданное программное средство;
* выбрана методика испытаний;
* описан процесс тестирования;

Проект был разработан в среде Microsoft Visual Studio 2022 на языке C#. База данных была разработана в системе управления базами данных Microsoft SQL Server 2019 с использованием технологии ADO.NET.

В ходе разработки была исследована предметная область, после чего осуществлялась проектировка базы данных, разработка структура приложения, дизайн Windows-form. Также был реализован ряд задач: просмотр списка грузов, клиентов, водителей, транспорта, городов и заявок, разграничение прав доступа к данным, добавление и редактирование информации об грузах, клиентах, транспорте, водителей, городах и заявках, формирование заявок, фильтрация списка грузов, просмотр списка оформленных заявок, изменение данных учетной записи, формирование документов по заявкам. Для проверки функционирования были составлены тесты, которые были успешно пройдены.

Разработка имеет интуитивно понятный графический интерфейс, позволяющий провести автоматизированную работу с данными о заявках.

В итоге, программное средство имеет свои преимущества:

* интуитивно понятный графический интерфейс;
* простота в использовании;
* отсутствие необходимости в обладании специальными навыками при работе с программой;
* создание документов на основании заявок.

# Список информационных источников

Бек, К Шаблоны реализации корпоративных приложений: учеб. пособие / К. Бек – М. : Вильямс, 2012. – 369 cтр.

Веймаер, Р. Освой самостоятельно Microsoft SQL Server 2000 за 21 день (+ CD–ROM): учеб. пособие / Р. Веймаер – М. : Вильямс, 2013. – 549 стр.

Голицына, О.Л. Базы данных: учеб. пособие / О. Л. Голицына Инфра – М, 2013. – 399 стр.

Дейт, К. Дж. Введение в системы баз данных: учеб. пособие / К. Дж. Дейт. К. : Диалектика – Издание 6, 2012. – 360 стр.

Дэвидсон, Луис проектирование баз данных на SQL Server 2000: учеб. пособие / Луис Дэвидсон – Бином, 2012. – 631 стр.

Дюваль, Поль М. Непрерывная интеграция. Улучшение качества программного обеспечения и снижение риска: учеб. пособие / Поль М Дюваль – М. : Вильямс, 2014. – 497 стр.

Наумов, А.Н. Системы управления базами данных и знаний: учеб. пособие / Наумов, А.Н.; Вендров, А.М.; Иванов, В.К. – М. : Финансы и статистика, 2012. – 352 стр.

Озкарахан, Э. Машины баз данных и управление базами данных: учеб. пособие / Э. Озкарахан – М. : Мир, 2013. – 551 стр.

Постолит, Анатолий Visual Studio .NET: разработка приложений баз данных: учеб. пособие / Анатолий Постолит – СПб: БХВ, 2013. – 544 стр.

Редько, В.Н. Базы данных и информационные системы: учеб. пособие / Редько, В.Н.; Бассараб, И.А. – Знание, 2012. – 602 стр.

# Приложение А (обязательное) Текст программы

«AdminPanel.cs»

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.SqlClient;

namespace YchetGryzoperevozok

{

public partial class AdminPanel : Form

{

private SqlConnection sqlConnection;

private SqlCommandBuilder sqlCommandBuilder;

private SqlDataAdapter sqlDataAdapter;

private DataSet dataSet;

private bool newRowAdding = false;

private string table = "EmployeesTable";

private int columnCount = 4;

public AdminPanel()

{

InitializeComponent();

using (StreamReader reader = new StreamReader("Name.txt"))

{

string text = reader.ReadToEnd();

lName.Text = text;

}

}

private void LoadData()

{

try

{

sqlDataAdapter = new SqlDataAdapter($"SELECT \*, N'Удалить' AS [Delete] FROM {table}", sqlConnection);

sqlCommandBuilder = new SqlCommandBuilder(sqlDataAdapter);

sqlCommandBuilder.GetInsertCommand();

sqlCommandBuilder.GetUpdateCommand();

sqlCommandBuilder.GetDeleteCommand();

dataSet = new DataSet();

sqlDataAdapter.Fill(dataSet, table);

dgvAdmin.DataSource = dataSet.Tables[table];

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

{

DataGridViewLinkCell linkCell = new DataGridViewLinkCell();

dgvAdmin[columnCount, i] = linkCell;

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void ReloadData()

{

try

{

dataSet.Tables[table].Clear();

sqlDataAdapter.Fill(dataSet, table);

dgvAdmin.DataSource = dataSet.Tables[table];

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

{

DataGridViewLinkCell linkCell = new DataGridViewLinkCell();

dgvAdmin[columnCount, i] = linkCell;

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

Authorization authorization = new Authorization();

authorization.Show();

}

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

{

sqlConnection = new SqlConnection(@"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\User\Desktop\Курсач\YchetGryzoperevozok\YchetGryzoperevozok\AuthorizationBD.mdf;Integrated Security=True");

sqlConnection.Open();

LoadData();

}

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

{

ReloadData();

}

private void dgvAdmin\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

try

{

if (e.ColumnIndex == columnCount)

{

string task = dgvAdmin.Rows[e.RowIndex].Cells[columnCount].Value.ToString();

if(task == "Удалить")

{

if(MessageBox.Show("Удалить эту строку ?", "Удаление", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)

{

int rowIndex = e.RowIndex;

dgvAdmin.Rows.RemoveAt(rowIndex);

dataSet.Tables[table].Rows[rowIndex].Delete();

sqlDataAdapter.Update(dataSet, table);

}

}

else if(task == "Добавить")

{

int rowIndex = dgvAdmin.Rows.Count - 2;

DataRow row = dataSet.Tables[table].NewRow();

row["Fio"] = dgvAdmin.Rows[rowIndex].Cells["Fio"].Value;

row["Password"] = dgvAdmin.Rows[rowIndex].Cells["Password"].Value;

row["Position"] = dgvAdmin.Rows[rowIndex].Cells["Position"].Value;

dataSet.Tables[table].Rows.Add(row);

dataSet.Tables[table].Rows.RemoveAt(dataSet.Tables[table].Rows.Count - 1);

dgvAdmin.Rows.RemoveAt(dgvAdmin.Rows.Count - 2);

dgvAdmin.Rows[e.RowIndex].Cells[4].Value = "Удалить";

sqlDataAdapter.Update(dataSet, table);

newRowAdding = false;

}

else if (task == "Обновить")

{

int r = e.RowIndex;

dataSet.Tables[table].Rows[r]["Fio"] = dgvAdmin.Rows[r].Cells["Fio"].Value;

dataSet.Tables[table].Rows[r]["Password"] = dgvAdmin.Rows[r].Cells["Password"].Value;

dataSet.Tables[table].Rows[r]["Position"] = dgvAdmin.Rows[r].Cells["Position"].Value;

sqlDataAdapter.Update(dataSet, table);

dgvAdmin.Rows[e.RowIndex].Cells[columnCount].Value = "Удалить";

}

ReloadData();

}

}

catch(Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void dgvAdmin\_UserAddedRow(object sender, DataGridViewRowEventArgs e)

{

try

{

if (newRowAdding == false)

{

newRowAdding = true;

int lastRow = dgvAdmin.Rows.Count - 2;

DataGridViewRow row = dgvAdmin.Rows[lastRow];

DataGridViewLinkCell linkCell = new DataGridViewLinkCell();

dgvAdmin[columnCount, lastRow] = linkCell;

row.Cells["Delete"].Value = "Добавить";

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void dgvAdmin\_CellValueChanged(object sender, DataGridViewCellEventArgs e)

{

try

{

if (newRowAdding == false)

{

int rowIndex = dgvAdmin.SelectedCells[0].RowIndex;

DataGridViewRow editingRow = dgvAdmin.Rows[rowIndex];

DataGridViewLinkCell linkCell = new DataGridViewLinkCell();

dgvAdmin[columnCount, rowIndex] = linkCell;

editingRow.Cells["Delete"].Value = "Обновить";

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

this.Close();

Authorization authorization = new Authorization();

authorization.Show();

}

private void dgvAdmin\_CellClick(object sender, DataGridViewCellEventArgs e)

{

int rowIndex = e.RowIndex;

label1.Text = Convert.ToString(rowIndex);

}

}

}

«Authorization.cs»

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace YchetGryzoperevozok

{

public partial class Authorization : Form

{

private SqlConnection sqlConnection;

private SqlDataReader sqlDataReader;

private SqlCommand command;

public Authorization()

{

InitializeComponent();

}

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

{

sqlConnection = new SqlConnection(@"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\User\Desktop\Курсач\YchetGryzoperevozok\YchetGryzoperevozok\AuthorizationBD.mdf;Integrated Security=True");

sqlConnection.Open();

command = new SqlCommand($"SELECT Position FROM PositionTable", sqlConnection);

sqlDataReader = command.ExecuteReader();

List<string> list = new List<string>();

while (sqlDataReader.Read())

{

list.Add(Convert.ToString(sqlDataReader["Position"]));

}

sqlDataReader.Close();

cbPosition.Items.AddRange(list.ToArray());

list.Clear();

command = new SqlCommand($"SELECT Fio FROM EmployeesTable", sqlConnection);

sqlDataReader = command.ExecuteReader();

while (sqlDataReader.Read())

{

list.Add(Convert.ToString(sqlDataReader["Fio"]));

}

sqlDataReader.Close();

cbName.Items.AddRange(list.ToArray());

}

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

{

try

{

if(cbName.Text != "" && tbInputPass.Text != "" && cbPosition.Text != "")

{

string login = cbName.Text;

string password = tbInputPass.Text;

string position = cbPosition.Text;

File.WriteAllText("Name.txt", login);

command = new SqlCommand($"SELECT \* FROM EmployeesTable WHERE Fio LIKE N'{login.ToUpper()}' AND Password LIKE N'{password.ToUpper()}' AND Position LIKE N'{position.ToUpper()}'", sqlConnection);

sqlDataReader = command.ExecuteReader();

if(sqlDataReader.HasRows)

{

while (sqlDataReader.Read())

{

if(position == "Сотрудник")

{

MainWindow mainWindow = new MainWindow();

mainWindow.Show();

}

else

{

AdminPanel adminPanel = new AdminPanel();

adminPanel.Show();

}

lEror.Text = "";

this.Hide();

}

sqlDataReader.Close();

}

else

{

lEror.Text = "Сотрудник не найден!";

tbInputPass.Clear();

sqlDataReader.Close();

}

}

else

{

lEror.Text = "Пустые поля!";

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

Application.Exit();

sqlConnection.Close();

}

}

}

«CargoTablePanel.cs»

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace YchetGryzoperevozok

{

public partial class CargoTablePanel : Form

{

private SqlConnection sqlConnection;

private SqlCommandBuilder sqlCommandBuilder;

private SqlDataAdapter sqlDataAdapter;

private DataSet dataSet;

private bool newRowAdding = false;

private string table = "CargoTable";

private int columnCount = 6;

public CargoTablePanel()

{

InitializeComponent();

using (StreamReader reader = new StreamReader("Name.txt"))

{

string text = reader.ReadToEnd();

lName.Text = text;

}

}

private void LoadData()

{

try

{

sqlDataAdapter = new SqlDataAdapter($"SELECT \*, N'Удалить' AS [Delete] FROM {table}", sqlConnection);

sqlCommandBuilder = new SqlCommandBuilder(sqlDataAdapter);

sqlCommandBuilder.GetInsertCommand();

sqlCommandBuilder.GetUpdateCommand();

sqlCommandBuilder.GetDeleteCommand();

dataSet = new DataSet();

sqlDataAdapter.Fill(dataSet, table);

dgvCargo.DataSource = dataSet.Tables[table];

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

{

DataGridViewLinkCell linkCell = new DataGridViewLinkCell();

dgvCargo[columnCount, i] = linkCell;

}

dgvCargo.Columns[0].HeaderText = "ID груза";

dgvCargo.Columns[1].HeaderText = "Название груза";

dgvCargo.Columns[2].HeaderText = "Вес, кг.";

dgvCargo.Columns[3].HeaderText = "Количество, шт.";

dgvCargo.Columns[4].HeaderText = "Цена, руб.";

dgvCargo.Columns[5].HeaderText = "ID заявки";

dgvCargo.Columns[6].HeaderText = "Функция";

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void ReloadData()

{

try

{

dataSet.Tables[table].Clear();

sqlDataAdapter.Fill(dataSet, table);

dgvCargo.DataSource = dataSet.Tables[table];

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

{

DataGridViewLinkCell linkCell = new DataGridViewLinkCell();

dgvCargo[columnCount, i] = linkCell;

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

this.Close();

}

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

{

sqlConnection = new SqlConnection(@"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\User\Desktop\Курсач\YchetGryzoperevozok\YchetGryzoperevozok\Gryzoperevozki.mdf;Integrated Security=True");

sqlConnection.Open();

LoadData();

}

private void dgvCargo\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

try

{

if (e.ColumnIndex == columnCount)

{

string task = dgvCargo.Rows[e.RowIndex].Cells[columnCount].Value.ToString();

if (task == "Удалить")

{

if (MessageBox.Show("Удалить эту строку ?", "Удаление", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)

{

int rowIndex = e.RowIndex;

dgvCargo.Rows.RemoveAt(rowIndex);

dataSet.Tables[table].Rows[rowIndex].Delete();

sqlDataAdapter.Update(dataSet, table);

}

}

else if (task == "Добавить")

{

int rowIndex = dgvCargo.Rows.Count - 2;

DataRow row = dataSet.Tables[table].NewRow();

row["Name of cargo"] = dgvCargo.Rows[rowIndex].Cells["Name of cargo"].Value;

row["Weight"] = dgvCargo.Rows[rowIndex].Cells["Weight"].Value;

row["Count"] = dgvCargo.Rows[rowIndex].Cells["Count"].Value;

row["Price"] = dgvCargo.Rows[rowIndex].Cells["Price"].Value;

row["id transportation"] = dgvCargo.Rows[rowIndex].Cells["id transportation"].Value;

dataSet.Tables[table].Rows.Add(row);

dataSet.Tables[table].Rows.RemoveAt(dataSet.Tables[table].Rows.Count - 1);

dgvCargo.Rows.RemoveAt(dgvCargo.Rows.Count - 2);

dgvCargo.Rows[e.RowIndex].Cells[columnCount].Value = "Удалить";

sqlDataAdapter.Update(dataSet, table);

newRowAdding = false;

}

else if (task == "Обновить")

{

int r = e.RowIndex;

dataSet.Tables[table].Rows[r]["Name of cargo"] = dgvCargo.Rows[r].Cells["Name of cargo"].Value;

dataSet.Tables[table].Rows[r]["Weight"] = dgvCargo.Rows[r].Cells["Weight"].Value;

dataSet.Tables[table].Rows[r]["Count"] = dgvCargo.Rows[r].Cells["Count"].Value;

dataSet.Tables[table].Rows[r]["Price"] = dgvCargo.Rows[r].Cells["Price"].Value;

dataSet.Tables[table].Rows[r]["id transportation"] = dgvCargo.Rows[r].Cells["id transportation"].Value;

sqlDataAdapter.Update(dataSet, table);

dgvCargo.Rows[e.RowIndex].Cells[columnCount].Value = "Удалить";

}

ReloadData();

}

}

catch (SqlException ex)

{

MessageBox.Show("Значение уже используется в другой таблице, поэтому не может быть удалено или изменено!", "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void dgvCargo\_UserAddedRow(object sender, DataGridViewRowEventArgs e)

{

try

{

if (newRowAdding == false)

{

newRowAdding = true;

int lastRow = dgvCargo.Rows.Count - 2;

DataGridViewRow row = dgvCargo.Rows[lastRow];

DataGridViewLinkCell linkCell = new DataGridViewLinkCell();

dgvCargo[columnCount, lastRow] = linkCell;

row.Cells["Delete"].Value = "Добавить";

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void dgvCargo\_CellValueChanged(object sender, DataGridViewCellEventArgs e)

{

try

{

if (newRowAdding == false)

{

int rowIndex = dgvCargo.SelectedCells[0].RowIndex;

DataGridViewRow editingRow = dgvCargo.Rows[rowIndex];

DataGridViewLinkCell linkCell = new DataGridViewLinkCell();

dgvCargo[columnCount, rowIndex] = linkCell;

editingRow.Cells["Delete"].Value = "Обновить";

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

ReloadData();

}

private void dgvCargo\_EditingControlShowing(object sender, DataGridViewEditingControlShowingEventArgs e)

{

e.Control.KeyPress -= new KeyPressEventHandler(Column\_KeyPress);

if(dgvCargo.CurrentCell.ColumnIndex == 2 || dgvCargo.CurrentCell.ColumnIndex == 3 || dgvCargo.CurrentCell.ColumnIndex == 4 || dgvCargo.CurrentCell.ColumnIndex == 5)

{

TextBox textBox = e.Control as TextBox;

if(textBox != null)

{

textBox.KeyPress += new KeyPressEventHandler(Column\_KeyPress);

}

}

}

private void Column\_KeyPress(object sender, KeyPressEventArgs e)

{

if(!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar))

{

e.Handled = true;

}

}

}

}

«CityTablePanel.cs»

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace YchetGryzoperevozok

{

public partial class CityTablePanel : Form

{

private SqlConnection sqlConnection;

private SqlCommandBuilder sqlCommandBuilder;

private SqlDataAdapter sqlDataAdapter;

private DataSet dataSet;

private bool newRowAdding = false;

private string table = "CityTable";

private int columnCount = 2;

public CityTablePanel()

{

InitializeComponent();

using (StreamReader reader = new StreamReader("Name.txt"))

{

string text = reader.ReadToEnd();

lName.Text = text;

}

}

private void LoadData()

{

try

{

sqlDataAdapter = new SqlDataAdapter($"SELECT \*, N'Удалить' AS [Delete] FROM {table}", sqlConnection);

sqlCommandBuilder = new SqlCommandBuilder(sqlDataAdapter);

sqlCommandBuilder.GetInsertCommand();

sqlCommandBuilder.GetUpdateCommand();

sqlCommandBuilder.GetDeleteCommand();

dataSet = new DataSet();

sqlDataAdapter.Fill(dataSet, table);

dgvCity.DataSource = dataSet.Tables[table];

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

{

DataGridViewLinkCell linkCell = new DataGridViewLinkCell();

dgvCity[columnCount, i] = linkCell;

}

dgvCity.Columns[0].HeaderText = "ID города";

dgvCity.Columns[1].HeaderText = "Название города";

dgvCity.Columns[2].HeaderText = "Функция";

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void ReloadData()

{

try

{

dataSet.Tables[table].Clear();

sqlDataAdapter.Fill(dataSet, table);

dgvCity.DataSource = dataSet.Tables[table];

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

{

DataGridViewLinkCell linkCell = new DataGridViewLinkCell();

dgvCity[columnCount, i] = linkCell;

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

this.Close();

}

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

{

sqlConnection = new SqlConnection(@"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\User\Desktop\Курсач\YchetGryzoperevozok\YchetGryzoperevozok\Gryzoperevozki.mdf;Integrated Security=True");

sqlConnection.Open();

LoadData();

}

private void dgvCity\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

try

{

if (e.ColumnIndex == columnCount)

{

string task = dgvCity.Rows[e.RowIndex].Cells[columnCount].Value.ToString();

if (task == "Удалить")

{

if (MessageBox.Show("Удалить эту строку ?", "Удаление", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)

{

int rowIndex = e.RowIndex;

dgvCity.Rows.RemoveAt(rowIndex);

dataSet.Tables[table].Rows[rowIndex].Delete();

sqlDataAdapter.Update(dataSet, table);

}

}

else if (task == "Добавить")

{

int rowIndex = dgvCity.Rows.Count - 2;

DataRow row = dataSet.Tables[table].NewRow();

row["Name of city"] = dgvCity.Rows[rowIndex].Cells["Name of city"].Value;

dataSet.Tables[table].Rows.Add(row);

dataSet.Tables[table].Rows.RemoveAt(dataSet.Tables[table].Rows.Count - 1);

dgvCity.Rows.RemoveAt(dgvCity.Rows.Count - 2);

dgvCity.Rows[e.RowIndex].Cells[columnCount].Value = "Удалить";

sqlDataAdapter.Update(dataSet, table);

newRowAdding = false;

}

else if (task == "Обновить")

{

int r = e.RowIndex;

dataSet.Tables[table].Rows[r]["Name of city"] = dgvCity.Rows[r].Cells["Name of city"].Value;

sqlDataAdapter.Update(dataSet, table);

dgvCity.Rows[e.RowIndex].Cells[columnCount].Value = "Удалить";

}

ReloadData();

}

}

catch (SqlException ex)

{

MessageBox.Show("Значение уже используется в другой таблице, поэтому не может быть удалено или изменено!", "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void dgvCity\_UserAddedRow(object sender, DataGridViewRowEventArgs e)

{

try

{

if (newRowAdding == false)

{

newRowAdding = true;

int lastRow = dgvCity.Rows.Count - 2;

DataGridViewRow row = dgvCity.Rows[lastRow];

DataGridViewLinkCell linkCell = new DataGridViewLinkCell();

dgvCity[columnCount, lastRow] = linkCell;

row.Cells["Delete"].Value = "Добавить";

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void dgvCity\_CellValueChanged(object sender, DataGridViewCellEventArgs e)

{

try

{

if (newRowAdding == false)

{

int rowIndex = dgvCity.SelectedCells[0].RowIndex;

DataGridViewRow editingRow = dgvCity.Rows[rowIndex];

DataGridViewLinkCell linkCell = new DataGridViewLinkCell();

dgvCity[columnCount, rowIndex] = linkCell;

editingRow.Cells["Delete"].Value = "Обновить";

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

ReloadData();

}

}

}

« ClientTablePanel.cs»

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace YchetGryzoperevozok

{

public partial class ClientTablePanel : Form

{

private SqlConnection sqlConnection;

private SqlCommandBuilder sqlCommandBuilder;

private SqlDataAdapter sqlDataAdapter;

private DataSet dataSet;

private bool newRowAdding = false;

private string table = "ClientTable";

private int columnCount = 5;

public ClientTablePanel()

{

InitializeComponent();

using (StreamReader reader = new StreamReader("Name.txt"))

{

string text = reader.ReadToEnd();

lName.Text = text;

}

}

private void LoadData()

{

try

{

sqlDataAdapter = new SqlDataAdapter($"SELECT \*, N'Удалить' AS [Delete] FROM {table}", sqlConnection);

sqlCommandBuilder = new SqlCommandBuilder(sqlDataAdapter);

sqlCommandBuilder.GetInsertCommand();

sqlCommandBuilder.GetUpdateCommand();

sqlCommandBuilder.GetDeleteCommand();

dataSet = new DataSet();

sqlDataAdapter.Fill(dataSet, table);

dgvClient.DataSource = dataSet.Tables[table];

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

{

DataGridViewLinkCell linkCell = new DataGridViewLinkCell();

dgvClient[columnCount, i] = linkCell;

}

dgvClient.Columns[0].HeaderText = "ID организации";

dgvClient.Columns[1].HeaderText = "Название организации";

dgvClient.Columns[2].HeaderText = "Телефон";

dgvClient.Columns[3].HeaderText = "Электронная почта";

dgvClient.Columns[4].HeaderText = "Улица";

dgvClient.Columns[5].HeaderText = "Функция";

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void ReloadData()

{

try

{

dataSet.Tables[table].Clear();

sqlDataAdapter.Fill(dataSet, table);

dgvClient.DataSource = dataSet.Tables[table];

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

{

DataGridViewLinkCell linkCell = new DataGridViewLinkCell();

dgvClient[columnCount, i] = linkCell;

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

this.Close();

}

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

{

sqlConnection = new SqlConnection(@"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\User\Desktop\Курсач\YchetGryzoperevozok\YchetGryzoperevozok\Gryzoperevozki.mdf;Integrated Security=True");

sqlConnection.Open();

LoadData();

}

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

{

ReloadData();

}

private void dgvClient\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

try

{

if (e.ColumnIndex == columnCount)

{

string task = dgvClient.Rows[e.RowIndex].Cells[columnCount].Value.ToString();

if (task == "Удалить")

{

if (MessageBox.Show("Удалить эту строку ?", "Удаление", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)

{

int rowIndex = e.RowIndex;

dgvClient.Rows.RemoveAt(rowIndex);

dataSet.Tables[table].Rows[rowIndex].Delete();

sqlDataAdapter.Update(dataSet, table);

}

}

else if (task == "Добавить")

{

int rowIndex = dgvClient.Rows.Count - 2;

DataRow row = dataSet.Tables[table].NewRow();

row["Name of org"] = dgvClient.Rows[rowIndex].Cells["Name of org"].Value;

row["Telephone"] = dgvClient.Rows[rowIndex].Cells["Telephone"].Value;

row["Email"] = dgvClient.Rows[rowIndex].Cells["Email"].Value;

row["Street"] = dgvClient.Rows[rowIndex].Cells["Street"].Value;

dataSet.Tables[table].Rows.Add(row);

dataSet.Tables[table].Rows.RemoveAt(dataSet.Tables[table].Rows.Count - 1);

dgvClient.Rows.RemoveAt(dgvClient.Rows.Count - 2);

dgvClient.Rows[e.RowIndex].Cells[columnCount].Value = "Удалить";

sqlDataAdapter.Update(dataSet, table);

newRowAdding = false;

}

else if (task == "Обновить")

{

int r = e.RowIndex;

dataSet.Tables[table].Rows[r]["Name of org"] = dgvClient.Rows[r].Cells["Name of org"].Value;

dataSet.Tables[table].Rows[r]["Telephone"] = dgvClient.Rows[r].Cells["Telephone"].Value;

dataSet.Tables[table].Rows[r]["Email"] = dgvClient.Rows[r].Cells["Email"].Value;

dataSet.Tables[table].Rows[r]["Street"] = dgvClient.Rows[r].Cells["Street"].Value;

sqlDataAdapter.Update(dataSet, table);

dgvClient.Rows[e.RowIndex].Cells[columnCount].Value = "Удалить";

}

ReloadData();

}

}

catch (SqlException ex)

{

MessageBox.Show("Значение уже используется в другой таблице, поэтому не может быть удалено или изменено!", "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void dgvClient\_UserAddedRow(object sender, DataGridViewRowEventArgs e)

{

try

{

if (newRowAdding == false)

{

newRowAdding = true;

int lastRow = dgvClient.Rows.Count - 2;

DataGridViewRow row = dgvClient.Rows[lastRow];

DataGridViewLinkCell linkCell = new DataGridViewLinkCell();

dgvClient[columnCount, lastRow] = linkCell;

row.Cells["Delete"].Value = "Добавить";

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void dgvClient\_CellValueChanged(object sender, DataGridViewCellEventArgs e)

{

try

{

if (newRowAdding == false)

{

int rowIndex = dgvClient.SelectedCells[0].RowIndex;

DataGridViewRow editingRow = dgvClient.Rows[rowIndex];

DataGridViewLinkCell linkCell = new DataGridViewLinkCell();

dgvClient[columnCount, rowIndex] = linkCell;

editingRow.Cells["Delete"].Value = "Обновить";

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

« DriverTablePanel.cs»

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace YchetGryzoperevozok

{

public partial class DriverTablePanel : Form

{

private SqlConnection sqlConnection;

private SqlCommandBuilder sqlCommandBuilder;

private SqlDataAdapter sqlDataAdapter;

private DataSet dataSet;

private bool newRowAdding = false;

private string table = "DriverTable";

private int columnCount = 2;

public DriverTablePanel()

{

InitializeComponent();

using (StreamReader reader = new StreamReader("Name.txt"))

{

string text = reader.ReadToEnd();

lName.Text = text;

}

}

private void LoadData()

{

try

{

sqlDataAdapter = new SqlDataAdapter($"SELECT \*, N'Удалить' AS [Delete] FROM {table}", sqlConnection);

sqlCommandBuilder = new SqlCommandBuilder(sqlDataAdapter);

sqlCommandBuilder.GetInsertCommand();

sqlCommandBuilder.GetUpdateCommand();

sqlCommandBuilder.GetDeleteCommand();

dataSet = new DataSet();

sqlDataAdapter.Fill(dataSet, table);

dgvDriver.DataSource = dataSet.Tables[table];

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

{

DataGridViewLinkCell linkCell = new DataGridViewLinkCell();

dgvDriver[columnCount, i] = linkCell;

}

dgvDriver.Columns[0].HeaderText = "ID водителя";

dgvDriver.Columns[1].HeaderText = "ФИО";

dgvDriver.Columns[2].HeaderText = "Функция";

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void ReloadData()

{

try

{

dataSet.Tables[table].Clear();

sqlDataAdapter.Fill(dataSet, table);

dgvDriver.DataSource = dataSet.Tables[table];

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

{

DataGridViewLinkCell linkCell = new DataGridViewLinkCell();

dgvDriver[columnCount, i] = linkCell;

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

this.Close();

}

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

{

sqlConnection = new SqlConnection(@"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\User\Desktop\Курсач\YchetGryzoperevozok\YchetGryzoperevozok\Gryzoperevozki.mdf;Integrated Security=True");

sqlConnection.Open();

LoadData();

}

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

{

ReloadData();

}

private void dgvDriver\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

try

{

if (e.ColumnIndex == columnCount)

{

string task = dgvDriver.Rows[e.RowIndex].Cells[columnCount].Value.ToString();

if (task == "Удалить")

{

if (MessageBox.Show("Удалить эту строку ?", "Удаление", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)

{

int rowIndex = e.RowIndex;

dgvDriver.Rows.RemoveAt(rowIndex);

dataSet.Tables[table].Rows[rowIndex].Delete();

sqlDataAdapter.Update(dataSet, table);

}

}

else if (task == "Добавить")

{

int rowIndex = dgvDriver.Rows.Count - 2;

DataRow row = dataSet.Tables[table].NewRow();

row["Fio"] = dgvDriver.Rows[rowIndex].Cells["Fio"].Value;

dataSet.Tables[table].Rows.Add(row);

dataSet.Tables[table].Rows.RemoveAt(dataSet.Tables[table].Rows.Count - 1);

dgvDriver.Rows.RemoveAt(dgvDriver.Rows.Count - 2);

dgvDriver.Rows[e.RowIndex].Cells[columnCount].Value = "Удалить";

sqlDataAdapter.Update(dataSet, table);

newRowAdding = false;

}

else if (task == "Обновить")

{

int r = e.RowIndex;

dataSet.Tables[table].Rows[r]["Fio"] = dgvDriver.Rows[r].Cells["Fio"].Value;

sqlDataAdapter.Update(dataSet, table);

dgvDriver.Rows[e.RowIndex].Cells[columnCount].Value = "Удалить";

}

ReloadData();

}

}

catch (SqlException ex)

{

MessageBox.Show("Значение уже используется в другой таблице, поэтому не может быть удалено или изменено!", "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void dgvDriver\_UserAddedRow(object sender, DataGridViewRowEventArgs e)

{

try

{

if (newRowAdding == false)

{

newRowAdding = true;

int lastRow = dgvDriver.Rows.Count - 2;

DataGridViewRow row = dgvDriver.Rows[lastRow];

DataGridViewLinkCell linkCell = new DataGridViewLinkCell();

dgvDriver[columnCount, lastRow] = linkCell;

row.Cells["Delete"].Value = "Добавить";

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void dgvDriver\_CellValueChanged(object sender, DataGridViewCellEventArgs e)

{

try

{

if (newRowAdding == false)

{

int rowIndex = dgvDriver.SelectedCells[0].RowIndex;

DataGridViewRow editingRow = dgvDriver.Rows[rowIndex];

DataGridViewLinkCell linkCell = new DataGridViewLinkCell();

dgvDriver[columnCount, rowIndex] = linkCell;

editingRow.Cells["Delete"].Value = "Обновить";

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

«ExcelHelper.cs»

using Microsoft.Office.Interop.Excel;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Excel = Microsoft.Office.Interop.Excel;

namespace YchetGryzoperevozok

{

class ExcelHelper : IDisposable

{

private Excel.Application excel;

private Workbook workbook;

private string filePath;

public ExcelHelper()

{

excel = new Excel.Application();

}

public void Dispose()

{

try

{

workbook.Close();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

internal bool Open(string filePath)

{

try

{

if(File.Exists(filePath))

{

workbook = excel.Workbooks.Open(filePath);

}

else

{

workbook = excel.Workbooks.Add();

filePath = filePath;

}

return true;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

return false;

}

internal bool Set(string column, int row, object data)

{

try

{

((Excel.Worksheet) excel.ActiveSheet).Cells[row, column] = data;

return true;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

return false;

}

internal void Save(string SavePathNewFile)

{

if(!string.IsNullOrEmpty(SavePathNewFile))

{

workbook.SaveAs(SavePathNewFile);

SavePathNewFile = null;

}

else

{

workbook.Save();

}

}

}

}

«MainWindow.cs»

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Diagnostics;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using Excel = Microsoft.Office.Interop.Excel;

namespace YchetGryzoperevozok

{

public partial class MainWindow : Form

{

public MainWindow()

{

InitializeComponent();

using (StreamReader reader = new StreamReader("Name.txt"))

{

string text = reader.ReadToEnd();

lName.Text = text;

}

}

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

{

Authorization authorization = new Authorization();

authorization.Show();

}

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

{

try

{

OpenFileDialog ofd = new OpenFileDialog();

ofd.DefaultExt = "\*.xls;\*.xlsx";

ofd.Filter = "Microsoft Excel (\*.xls\*)|\*.xls\*";

ofd.Title = "Выберите документ Excel";

if (ofd.ShowDialog() != DialogResult.OK)

{

MessageBox.Show("Вы не выбрали файл для открытия", "Загрузка данных...", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

else

{

Process p = new Process();

p.StartInfo.UseShellExecute = true;

p.StartInfo.FileName = ofd.FileName;

p.Start();

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

TransportationsTablePanel transportationsTablePanel = new TransportationsTablePanel();

transportationsTablePanel.Show();

}

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

{

MessageBox.Show("О программе");

}

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

{

this.Close();

Authorization authorization = new Authorization();

authorization.Show();

}

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

{

ClientTablePanel authorization = new ClientTablePanel();

authorization.Show();

}

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

{

DriverTablePanel authorization = new DriverTablePanel();

authorization.Show();

}

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

{

TransportTablePanel authorization = new TransportTablePanel();

authorization.Show();

}

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

{

CityTablePanel authorization = new CityTablePanel();

authorization.Show();

}

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

{

CargoTablePanel authorization = new CargoTablePanel();

authorization.Show();

}

}

}

«TransportationsTablePanel.cs»

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using Excel = Microsoft.Office.Interop.Excel;

namespace YchetGryzoperevozok

{

public partial class TransportationsTablePanel : Form

{

private SqlConnection sqlConnection;

private SqlCommandBuilder sqlCommandBuilderCargo;

private SqlCommandBuilder sqlCommandBuilderTransportation;

private SqlDataAdapter sqlDataAdapterCargo;

private SqlDataAdapter sqlDataAdapterTransportation;

private DataSet dataSetCargo;

private DataSet dataSetTransportation;

private SqlCommand command;

private SqlDataReader sqlDataReader;

private bool newRowAddingCargo = false;

private bool newRowAddingTransportation = false;

private string tableCargo = "CargoTable";

private int columnCountCargo = 6;

private string tableTransportation = "TransportationTable";

private int columnCountTransportation = 5;

private int idTransportation = 1017;

public TransportationsTablePanel()

{

InitializeComponent();

using (StreamReader reader = new StreamReader("Name.txt"))

{

string text = reader.ReadToEnd();

lName.Text = text;

}

}

private void LoadDataCargo()

{

try

{

sqlDataAdapterCargo = new SqlDataAdapter($"SELECT \*, N'Удалить' AS [Delete] FROM {tableCargo} WHERE {tableCargo}.[id transportation] = {idTransportation}", sqlConnection);

sqlCommandBuilderCargo = new SqlCommandBuilder(sqlDataAdapterCargo);

sqlCommandBuilderCargo.GetInsertCommand();

sqlCommandBuilderCargo.GetUpdateCommand();

sqlCommandBuilderCargo.GetDeleteCommand();

dataSetCargo = new DataSet();

sqlDataAdapterCargo.Fill(dataSetCargo, tableCargo);

dgvCargo.DataSource = dataSetCargo.Tables[tableCargo];

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

{

DataGridViewLinkCell linkCell = new DataGridViewLinkCell();

dgvCargo[columnCountCargo, i] = linkCell;

}

dgvCargo.Columns[0].HeaderText = "ID груза";

dgvCargo.Columns[1].HeaderText = "Название груза";

dgvCargo.Columns[2].HeaderText = "Вес, кг.";

dgvCargo.Columns[3].HeaderText = "Количество, шт.";

dgvCargo.Columns[4].HeaderText = "Цена, руб.";

dgvCargo.Columns[5].HeaderText = "ID заявки";

dgvCargo.Columns[6].HeaderText = "Функция";

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void LoadDataTransportation()

{

try

{

sqlDataAdapterTransportation = new SqlDataAdapter($"SELECT \*, N'Удалить' AS [Delete] FROM {tableTransportation}", sqlConnection);

sqlCommandBuilderTransportation = new SqlCommandBuilder(sqlDataAdapterTransportation);

sqlCommandBuilderTransportation.GetInsertCommand();

sqlCommandBuilderTransportation.GetUpdateCommand();

sqlCommandBuilderTransportation.GetDeleteCommand();

dataSetTransportation = new DataSet();

sqlDataAdapterTransportation.Fill(dataSetTransportation, tableTransportation);

dgvTransportation.DataSource = dataSetTransportation.Tables[tableTransportation];

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

{

DataGridViewLinkCell linkCell = new DataGridViewLinkCell();

dgvTransportation[columnCountTransportation, i] = linkCell;

}

dgvTransportation.Columns[0].HeaderText = "ID заявки";

dgvTransportation.Columns[1].HeaderText = "Название организации";

dgvTransportation.Columns[2].HeaderText = "Номер автомобиля";

dgvTransportation.Columns[3].HeaderText = "Город";

dgvTransportation.Columns[4].HeaderText = "Дата";

dgvTransportation.Columns[5].HeaderText = "Функция";

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void ReloadDataCargo()

{

try

{

dataSetCargo.Tables[tableCargo].Clear();

sqlDataAdapterCargo.Fill(dataSetCargo, tableCargo);

dgvCargo.DataSource = dataSetCargo.Tables[tableCargo];

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

{

DataGridViewLinkCell linkCell = new DataGridViewLinkCell();

dgvCargo[columnCountCargo, i] = linkCell;

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void ReloadDataTransportation()

{

try

{

dataSetTransportation.Tables[tableTransportation].Clear();

sqlDataAdapterTransportation.Fill(dataSetTransportation, tableTransportation);

dgvTransportation.DataSource = dataSetTransportation.Tables[tableTransportation];

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

{

DataGridViewLinkCell linkCell = new DataGridViewLinkCell();

dgvTransportation[columnCountTransportation, i] = linkCell;

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

this.Close();

}

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

{

sqlConnection = new SqlConnection(@"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\User\Desktop\Курсач\YchetGryzoperevozok\YchetGryzoperevozok\Gryzoperevozki.mdf;Integrated Security=True");

sqlConnection.Open();

LoadDataCargo();

LoadDataTransportation();

string zapros = "SELECT [Name of cargo] FROM CargoTable";

string column = "Name of cargo";

List<string> list = ComboBoxInsert(sqlConnection, sqlDataReader, zapros, column);

cbNameOfCargo.Items.AddRange(list.ToArray());

list.Clear();

zapros = "SELECT [Weight] FROM CargoTable";

column = "Weight";

list = ComboBoxInsert(sqlConnection, sqlDataReader, zapros, column);

cbWeight.Items.AddRange(list.ToArray());

list.Clear();

zapros = "SELECT [Count] FROM CargoTable";

column = "Count";

list = ComboBoxInsert(sqlConnection, sqlDataReader, zapros, column);

cbCount.Items.AddRange(list.ToArray());

list.Clear();

zapros = "SELECT [Price] FROM CargoTable";

column = "Price";

list = ComboBoxInsert(sqlConnection, sqlDataReader, zapros, column);

cbPrice.Items.AddRange(list.ToArray());

list.Clear();

zapros = "SELECT [id transportation] FROM CargoTable";

column = "id transportation";

list = ComboBoxInsert(sqlConnection, sqlDataReader, zapros, column);

cbIdTransportation.Items.AddRange(list.ToArray());

list.Clear();

zapros = "SELECT [id cargo] FROM CargoTable";

column = "id cargo";

list = ComboBoxInsert(sqlConnection, sqlDataReader, zapros, column);

cbIdCargo.Items.AddRange(list.ToArray());

list.Clear();

zapros = "SELECT [id transportation] FROM TransportationTable";

column = "id transportation";

list = ComboBoxInsert(sqlConnection, sqlDataReader, zapros, column);

cbId.Items.AddRange(list.ToArray());

list.Clear();

zapros = "SELECT [Name of org] FROM ClientTable";

column = "Name of org";

list = ComboBoxInsert(sqlConnection, sqlDataReader, zapros, column);

cbNameOfOrg.Items.AddRange(list.ToArray());

list.Clear();

zapros = "SELECT [Name of city] FROM CityTable";

column = "Name of city";

list = ComboBoxInsert(sqlConnection, sqlDataReader, zapros, column);

cbNameOfCity.Items.AddRange(list.ToArray());

list.Clear();

zapros = "SELECT [Nomer] FROM TransportTable";

column = "Nomer";

list = ComboBoxInsert(sqlConnection, sqlDataReader, zapros, column);

cbNomer.Items.AddRange(list.ToArray());

list.Clear();

}

private List<string> ComboBoxInsert(SqlConnection sqlConnection, SqlDataReader sqlDataReader, string zapros, string column)

{

command = new SqlCommand(zapros, sqlConnection);

sqlDataReader = command.ExecuteReader();

List<string> list = new List<string>();

while (sqlDataReader.Read())

{

list.Add(Convert.ToString(sqlDataReader[column]));

}

sqlDataReader.Close();

return list;

}

private void dgvTransportation\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

try

{

if (e.ColumnIndex == columnCountTransportation)

{

string task = dgvTransportation.Rows[e.RowIndex].Cells[columnCountTransportation].Value.ToString();

if (task == "Удалить")

{

if (MessageBox.Show("Удалить эту строку ?", "Удаление", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)

{

int rowIndex = e.RowIndex;

dgvTransportation.Rows.RemoveAt(rowIndex);

dataSetTransportation.Tables[tableTransportation].Rows[rowIndex].Delete();

sqlDataAdapterTransportation.Update(dataSetTransportation, tableTransportation);

}

}

else if (task == "Добавить")

{

int rowIndex = dgvTransportation.Rows.Count - 2;

DataRow row = dataSetTransportation.Tables[tableTransportation].NewRow();

row["Name of org"] = dgvTransportation.Rows[rowIndex].Cells["Name of org"].Value;

row["Nomer"] = dgvTransportation.Rows[rowIndex].Cells["Nomer"].Value;

row["Name of city"] = dgvTransportation.Rows[rowIndex].Cells["Name of city"].Value;

row["Date"] = dgvTransportation.Rows[rowIndex].Cells["Date"].Value;

dataSetTransportation.Tables[tableTransportation].Rows.Add(row);

dataSetTransportation.Tables[tableTransportation].Rows.RemoveAt(dataSetTransportation.Tables[tableTransportation].Rows.Count - 1);

dgvTransportation.Rows.RemoveAt(dgvTransportation.Rows.Count - 2);

dgvTransportation.Rows[e.RowIndex].Cells[columnCountTransportation].Value = "Удалить";

sqlDataAdapterTransportation.Update(dataSetTransportation, tableTransportation);

newRowAddingTransportation = false;

}

else if (task == "Обновить")

{

int r = e.RowIndex;

dataSetTransportation.Tables[tableTransportation].Rows[r]["Name of org"] = dgvTransportation.Rows[r].Cells["Name of org"].Value;

dataSetTransportation.Tables[tableTransportation].Rows[r]["Nomer"] = dgvTransportation.Rows[r].Cells["Nomer"].Value;

dataSetTransportation.Tables[tableTransportation].Rows[r]["Name of city"] = dgvTransportation.Rows[r].Cells["Name of city"].Value;

dataSetTransportation.Tables[tableTransportation].Rows[r]["Date"] = dgvTransportation.Rows[r].Cells["Date"].Value;

sqlDataAdapterTransportation.Update(dataSetTransportation, tableTransportation);

dgvTransportation.Rows[e.RowIndex].Cells[columnCountTransportation].Value = "Удалить";

}

ReloadDataTransportation();

}

}

catch (SqlException ex)

{

MessageBox.Show("Значение уже используется в другой таблице, поэтому не может быть удалено или изменено!", "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void dgvCargo\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

try

{

if (e.ColumnIndex == columnCountCargo)

{

string task = dgvCargo.Rows[e.RowIndex].Cells[columnCountCargo].Value.ToString();

if (task == "Удалить")

{

if (MessageBox.Show("Удалить эту строку ?", "Удаление", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)

{

int rowIndex = e.RowIndex;

dgvCargo.Rows.RemoveAt(rowIndex);

dataSetCargo.Tables[tableCargo].Rows[rowIndex].Delete();

sqlDataAdapterCargo.Update(dataSetCargo, tableCargo);

}

}

else if (task == "Добавить")

{

int rowIndex = dgvCargo.Rows.Count - 2;

DataRow row = dataSetCargo.Tables[tableCargo].NewRow();

row["Name of cargo"] = dgvCargo.Rows[rowIndex].Cells["Name of cargo"].Value;

row["Weight"] = dgvCargo.Rows[rowIndex].Cells["Weight"].Value;

row["Count"] = dgvCargo.Rows[rowIndex].Cells["Count"].Value;

row["Price"] = dgvCargo.Rows[rowIndex].Cells["Price"].Value;

row["id transportation"] = idTransportation;

dataSetCargo.Tables[tableCargo].Rows.Add(row);

dataSetCargo.Tables[tableCargo].Rows.RemoveAt(dataSetCargo.Tables[tableCargo].Rows.Count - 1);

dgvCargo.Rows.RemoveAt(dgvCargo.Rows.Count - 2);

dgvCargo.Rows[e.RowIndex].Cells[columnCountCargo].Value = "Удалить";

sqlDataAdapterCargo.Update(dataSetCargo, tableCargo);

newRowAddingCargo = false;

}

else if (task == "Обновить")

{

int r = e.RowIndex;

dataSetCargo.Tables[tableCargo].Rows[r]["Name of cargo"] = dgvCargo.Rows[r].Cells["Name of cargo"].Value;

dataSetCargo.Tables[tableCargo].Rows[r]["Weight"] = dgvCargo.Rows[r].Cells["Weight"].Value;

dataSetCargo.Tables[tableCargo].Rows[r]["Count"] = dgvCargo.Rows[r].Cells["Count"].Value;

dataSetCargo.Tables[tableCargo].Rows[r]["Price"] = dgvCargo.Rows[r].Cells["Price"].Value;

dataSetCargo.Tables[tableCargo].Rows[r]["id transportation"] = idTransportation;

sqlDataAdapterCargo.Update(dataSetCargo, tableCargo);

dgvCargo.Rows[e.RowIndex].Cells[columnCountCargo].Value = "Удалить";

}

ReloadDataCargo();

}

}

catch (SqlException ex)

{

MessageBox.Show("Значение уже используется в другой таблице, поэтому не может быть удалено или изменено!", "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void dgvTransportation\_UserAddedRow(object sender, DataGridViewRowEventArgs e)

{

try

{

if (newRowAddingTransportation == false)

{

newRowAddingTransportation = true;

int lastRow = dgvTransportation.Rows.Count - 2;

DataGridViewRow row = dgvTransportation.Rows[lastRow];

DataGridViewLinkCell linkCell = new DataGridViewLinkCell();

dgvTransportation[columnCountTransportation, lastRow] = linkCell;

row.Cells["Delete"].Value = "Добавить";

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void dgvCargo\_UserAddedRow(object sender, DataGridViewRowEventArgs e)

{

try

{

if (newRowAddingCargo == false)

{

newRowAddingCargo = true;

int lastRow = dgvCargo.Rows.Count - 2;

DataGridViewRow row = dgvCargo.Rows[lastRow];

DataGridViewLinkCell linkCell = new DataGridViewLinkCell();

dgvCargo[columnCountCargo, lastRow] = linkCell;

row.Cells["Delete"].Value = "Добавить";

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void dgvTransportation\_CellValueChanged(object sender, DataGridViewCellEventArgs e)

{

try

{

if (newRowAddingTransportation == false)

{

int rowIndex = dgvTransportation.SelectedCells[0].RowIndex;

DataGridViewRow editingRow = dgvTransportation.Rows[rowIndex];

DataGridViewLinkCell linkCell = new DataGridViewLinkCell();

dgvTransportation[columnCountTransportation, rowIndex] = linkCell;

editingRow.Cells["Delete"].Value = "Обновить";

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void dgvCargo\_CellValueChanged(object sender, DataGridViewCellEventArgs e)

{

try

{

if (newRowAddingCargo == false)

{

int rowIndex = dgvCargo.SelectedCells[0].RowIndex;

DataGridViewRow editingRow = dgvCargo.Rows[rowIndex];

DataGridViewLinkCell linkCell = new DataGridViewLinkCell();

dgvCargo[columnCountCargo, rowIndex] = linkCell;

editingRow.Cells["Delete"].Value = "Обновить";

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

ReloadDataCargo();

ReloadDataTransportation();

}

private void dgvTransportation\_CellClick(object sender, DataGridViewCellEventArgs e)

{

try

{

int r = e.RowIndex;

string id = Convert.ToString(dgvTransportation.Rows[r].Cells["id transportation"].Value);

if (id != "")

{

idTransportation = Convert.ToInt32(dgvTransportation.Rows[r].Cells["id transportation"].Value);

LoadDataCargo();

cbNameOfOrg.Text = Convert.ToString(dgvTransportation.Rows[r].Cells["Name of org"].Value);

cbNomer.Text = Convert.ToString(dgvTransportation.Rows[r].Cells["Nomer"].Value);

cbNameOfCity.Text = Convert.ToString(dgvTransportation.Rows[r].Cells["Name of city"].Value);

dtpData.Value = Convert.ToDateTime(dgvTransportation.Rows[r].Cells["Date"].Value);

cbId.Text = Convert.ToString(idTransportation);

}

else

{

cbNameOfOrg.Text = "";

cbNomer.Text = "";

cbNameOfCity.Text = "";

cbId.Text = "";

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void dgvCargo\_CellClick(object sender, DataGridViewCellEventArgs e)

{

try

{

int r = e.RowIndex;

string id = Convert.ToString(dgvCargo.Rows[r].Cells["Id cargo"].Value);

if (id != "")

{

cbNameOfCargo.Text = Convert.ToString(dgvCargo.Rows[r].Cells["Name of cargo"].Value);

cbWeight.Text = Convert.ToString(dgvCargo.Rows[r].Cells["Weight"].Value);

cbCount.Text = Convert.ToString(dgvCargo.Rows[r].Cells["Count"].Value);

cbPrice.Text = Convert.ToString(dgvCargo.Rows[r].Cells["Price"].Value);

cbIdTransportation.Text = Convert.ToString(dgvCargo.Rows[r].Cells["id transportation"].Value);

cbIdCargo.Text = Convert.ToString(dgvCargo.Rows[r].Cells["id cargo"].Value);

}

else

{

cbNameOfCargo.Text = "";

cbWeight.Text = "";

cbCount.Text = "";

cbPrice.Text = "";

cbIdTransportation.Text = "";

cbIdCargo.Text = "";

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void dgvCargo\_EditingControlShowing(object sender, DataGridViewEditingControlShowingEventArgs e)

{

e.Control.KeyPress -= new KeyPressEventHandler(Column\_KeyPress);

if (dgvCargo.CurrentCell.ColumnIndex == 2 || dgvCargo.CurrentCell.ColumnIndex == 3 || dgvCargo.CurrentCell.ColumnIndex == 4 || dgvCargo.CurrentCell.ColumnIndex == 5)

{

TextBox textBox = e.Control as TextBox;

if (textBox != null)

{

textBox.KeyPress += new KeyPressEventHandler(Column\_KeyPress);

}

}

}

private void Column\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar))

{

e.Handled = true;

}

}

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

{

try

{

if (cbNameOfCity.Text != "" && cbNameOfOrg.Text != "" && cbNomer.Text != "")

{

lEror.Text = "";

SqlCommand command = new SqlCommand("INSERT INTO [TransportationTable] VALUES(@NameOfOrg, @Nomer, @NameOfCity, @Date)", sqlConnection);

command.Parameters.AddWithValue("NameOfOrg", cbNameOfOrg.Text);

command.Parameters.AddWithValue("Nomer", cbNomer.Text);

command.Parameters.AddWithValue("NameOfCity", cbNameOfCity.Text);

command.Parameters.AddWithValue("Date", dtpData.Value);

command.ExecuteNonQuery();

ReloadDataTransportation();

}

else

{

lEror.Text = "Пустые поля!";

}

}

catch (SqlException ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

try

{

if (cbNameOfCity.Text != "" && cbNameOfOrg.Text != "" && cbNomer.Text != "" && cbId.Text != "")

{

lEror.Text = "";

SqlCommand command = new SqlCommand("UPDATE [TransportationTable] SET [Name of org]=@NameOfOrg, [Nomer]=@Nomer, [Name of city]=@NameOfCity, [Date]=@Date WHERE [id transportation]=@Id", sqlConnection);

command.Parameters.AddWithValue("NameOfOrg", cbNameOfOrg.Text);

command.Parameters.AddWithValue("Nomer", cbNomer.Text);

command.Parameters.AddWithValue("NameOfCity", cbNameOfCity.Text);

command.Parameters.AddWithValue("Date", dtpData.Value);

command.Parameters.AddWithValue("Id", cbId.Text);

command.ExecuteNonQuery();

ReloadDataTransportation();

}

else

{

lEror.Text = "Пустые поля!";

}

}

catch (SqlException ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

try

{

if (cbId.Text != "")

{

lEror.Text = "";

SqlCommand command = new SqlCommand("DELETE FROM [TransportationTable] WHERE [id transportation]=@Id", sqlConnection);

command.Parameters.AddWithValue("Id", cbId.Text);

command.ExecuteNonQuery();

ReloadDataTransportation();

}

else

{

lEror.Text = "Пустое поле ID!";

}

}

catch (SqlException ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

try

{

if (cbNameOfCargo.Text != "" && cbWeight.Text != "" && cbCount.Text != "" && cbPrice.Text != "" && cbIdTransportation.Text != "")

{

lEror.Text = "";

SqlCommand command = new SqlCommand("INSERT INTO [CargoTable] VALUES(@NameOfCargo, @Weight, @Count, @Price, @IdTransportation)", sqlConnection);

command.Parameters.AddWithValue("NameOfCargo", cbNameOfCargo.Text);

command.Parameters.AddWithValue("Weight", cbWeight.Text);

command.Parameters.AddWithValue("Count", cbCount.Text);

command.Parameters.AddWithValue("Price", cbPrice.Text);

command.Parameters.AddWithValue("IdTransportation", cbIdTransportation.Text);

command.ExecuteNonQuery();

ReloadDataCargo();

}

else

{

lEror.Text = "Пустые поля!";

}

}

catch (SqlException ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

try

{

if (cbNameOfCargo.Text != "" && cbWeight.Text != "" && cbCount.Text != "" && cbPrice.Text != "" && cbIdTransportation.Text != "")

{

lEror.Text = "";

SqlCommand command = new SqlCommand("UPDATE [CargoTable] SET [Name of cargo]=@NameOfCargo, [Weight]=@Weight, [Count]=@Count, [Price]=@Price, [id transportation]=@IdTransportation WHERE [Id cargo]=@Id", sqlConnection);

command.Parameters.AddWithValue("NameOfCargo", cbNameOfCargo.Text);

command.Parameters.AddWithValue("Weight", cbWeight.Text);

command.Parameters.AddWithValue("Count", cbCount.Text);

command.Parameters.AddWithValue("Price", cbPrice.Text);

command.Parameters.AddWithValue("IdTransportation", cbIdTransportation.Text);

command.Parameters.AddWithValue("Id", cbIdCargo.Text);

command.ExecuteNonQuery();

ReloadDataCargo();

}

else

{

lEror.Text = "Пустые поля!";

}

}

catch (SqlException ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

try

{

if (cbIdCargo.Text != "")

{

lEror.Text = "";

SqlCommand command = new SqlCommand("DELETE FROM [CargoTable] WHERE [Id cargo]=@Id", sqlConnection);

command.Parameters.AddWithValue("Id", cbIdCargo.Text);

command.ExecuteNonQuery();

ReloadDataCargo();

}

else

{

lEror.Text = "Пустое поле ID!";

}

}

catch (SqlException ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

try

{

if (cbIdTransportation.Text != "" && cbNameOfOrg.Text != "" && cbNameOfCity.Text != "" && cbNomer.Text != "" && cbIdCargo.Text != "" && cbNameOfCargo.Text != "" && cbWeight.Text != "" && cbCount.Text != "" && cbPrice.Text != "")

{

SqlCommand sql = new SqlCommand($"SELECT [id driver] FROM TransportTable WHERE [Nomer]=N'{cbNomer.Text}';", sqlConnection);

int IdDriver = (int)sql.ExecuteScalar();

sql = new SqlCommand($"SELECT [Fio] FROM DriverTable WHERE [id driver]={IdDriver};", sqlConnection);

string FioDriver = (string)sql.ExecuteScalar();

sql = new SqlCommand($"SELECT [Name] FROM TransportTable WHERE [Nomer]=N'{cbNomer.Text}';", sqlConnection);

string Transport = (string)sql.ExecuteScalar();

using (ExcelHelper helper = new ExcelHelper())

{

if (helper.Open(filePath: Path.Combine(Environment.CurrentDirectory, "DogovorNaDostavky.xlsx")))

{

SaveFileDialog save = new SaveFileDialog();

save.Filter = "Excel document| \*.xlsx";

if (save.ShowDialog() == System.Windows.Forms.DialogResult.OK)

{

string zapros\_1 = $"SELECT [Name of cargo] FROM CargoTable WHERE [id transportation]={cbIdTransportation.Text};";

string column\_1 = "Name of cargo";

List<string> list = ComboBoxInsert(sqlConnection, sqlDataReader, zapros\_1, column\_1);

string zapros\_2 = $"SELECT [Weight] FROM CargoTable WHERE [id transportation]={cbIdTransportation.Text};";

string column\_2 = "Weight";

List<string> list\_2 = ComboBoxInsert(sqlConnection, sqlDataReader, zapros\_2, column\_2);

string zapros\_3 = $"SELECT [Count] FROM CargoTable WHERE [id transportation]={cbIdTransportation.Text};";

string column\_3 = "Count";

List<string> list\_3 = ComboBoxInsert(sqlConnection, sqlDataReader, zapros\_3, column\_3);

string zapros\_4 = $"SELECT [Price] FROM CargoTable WHERE [id transportation]={cbIdTransportation.Text};";

string column\_4 = "Price";

List<string> list\_4 = ComboBoxInsert(sqlConnection, sqlDataReader, zapros\_4, column\_4);

string SavePathNewFile = save.FileName;

helper.Set(column: "C", row: 7, data: Transport);

helper.Set(column: "C", row: 8, data: FioDriver);

helper.Set(column: "C", row: 10, data: cbNameOfOrg.Text);

int rowCount = 13, rowData = 0;

for (int i = 0; i < list.Count; i++)

{

helper.Set(column: "C", row: rowCount, data: list[i]);

helper.Set(column: "D", row: rowCount, data: list\_2[i]);

helper.Set(column: "E", row: rowCount, data: list\_3[i]);

helper.Set(column: "F", row: rowCount, data: list\_4[i]);

rowCount++;

rowData = rowCount + 1;

}

helper.Set(column: "C", row: rowData, data: cbNameOfCity.Text);

helper.Set(column: "B", row: rowData, data: "Место доставки");

helper.Set(column: "D", row: rowData + 2, data: lName.Text);

helper.Set(column: "B", row: rowData + 2, data: "Сотрудник");

helper.Set(column: "C", row: rowData + 2, data: "\_\_\_\_\_\_\_\_\_\_\_\_\_");

helper.Save(SavePathNewFile);

}

}

}

}

else

{

MessageBox.Show("Необходимые поля для договора на поставку пустые!", "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

try

{

if (cbIdTransportation.Text != "" && cbNameOfOrg.Text != "" && cbNameOfCity.Text != "" && cbNomer.Text != "" && cbIdCargo.Text != "" && cbNameOfCargo.Text != "" && cbWeight.Text != "" && cbCount.Text != "" && cbPrice.Text != "")

{

SqlCommand sql = new SqlCommand($"SELECT [id driver] FROM TransportTable WHERE [Nomer]=N'{cbNomer.Text}';", sqlConnection);

int IdDriver = (int)sql.ExecuteScalar();

sql = new SqlCommand($"SELECT [Fio] FROM DriverTable WHERE [id driver]={IdDriver};", sqlConnection);

string FioDriver = (string)sql.ExecuteScalar();

sql = new SqlCommand($"SELECT [Name] FROM TransportTable WHERE [Nomer]=N'{cbNomer.Text}';", sqlConnection);

string Transport = (string)sql.ExecuteScalar();

using (ExcelHelper helper = new ExcelHelper())

{

if (helper.Open(filePath: Path.Combine(Environment.CurrentDirectory, "PytevoiList.xlsx")))

{

SaveFileDialog save = new SaveFileDialog();

save.Filter = "Excel document| \*.xlsx";

if (save.ShowDialog() == System.Windows.Forms.DialogResult.OK)

{

string zapros\_1 = $"SELECT [Name of cargo] FROM CargoTable WHERE [id transportation]={cbIdTransportation.Text};";

string column\_1 = "Name of cargo";

List<string> list = ComboBoxInsert(sqlConnection, sqlDataReader, zapros\_1, column\_1);

string zapros\_2 = $"SELECT [Weight] FROM CargoTable WHERE [id transportation]={cbIdTransportation.Text};";

string column\_2 = "Weight";

List<string> list\_2 = ComboBoxInsert(sqlConnection, sqlDataReader, zapros\_2, column\_2);

string zapros\_3 = $"SELECT [Count] FROM CargoTable WHERE [id transportation]={cbIdTransportation.Text};";

string column\_3 = "Count";

List<string> list\_3 = ComboBoxInsert(sqlConnection, sqlDataReader, zapros\_3, column\_3);

string zapros\_4 = $"SELECT [Price] FROM CargoTable WHERE [id transportation]={cbIdTransportation.Text};";

string column\_4 = "Price";

List<string> list\_4 = ComboBoxInsert(sqlConnection, sqlDataReader, zapros\_4, column\_4);

helper.Set(column: "C", row: 8, data: Transport);

helper.Set(column: "C", row: 6, data: FioDriver);

helper.Set(column: "C", row: 9, data: cbNameOfOrg.Text);

helper.Set(column: "C", row: 10, data: cbNameOfCity.Text);

helper.Set(column: "C", row: 7, data: cbNomer.Text);

helper.Set(column: "E", row: 6, data: dtpData.Value);

string SavePathNewFile = save.FileName;

int rowCount = 15, rowData = 0;

for (int i = 0; i < list.Count; i++)

{

helper.Set(column: "B", row: rowCount, data: list[i]);

helper.Set(column: "C", row: rowCount, data: list\_2[i]);

helper.Set(column: "D", row: rowCount, data: list\_3[i]);

helper.Set(column: "E", row: rowCount, data: list\_4[i]);

rowCount++;

rowData = rowCount + 1;

}

helper.Set(column: "D", row: rowData + 2, data: lName.Text);

helper.Set(column: "B", row: rowData + 2, data: "Сотрудник");

helper.Set(column: "C", row: rowData + 2, data: "\_\_\_\_\_\_\_\_\_\_\_\_\_");

helper.Set(column: "F", row: rowData + 2, data: "Водитель");

helper.Set(column: "G", row: rowData + 2, data: "\_\_\_\_\_\_\_\_\_\_\_\_\_");

helper.Save(SavePathNewFile);

}

}

}

}

else

{

MessageBox.Show("Необходимые поля для договора на поставку пустые!", "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

«TransportTablePanel.cs»

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace YchetGryzoperevozok

{

public partial class TransportTablePanel : Form

{

private SqlConnection sqlConnection;

private SqlCommandBuilder sqlCommandBuilder;

private SqlDataAdapter sqlDataAdapter;

private DataSet dataSet;

private bool newRowAdding = false;

private string table = "TransportTable";

private int columnCount = 5;

public TransportTablePanel()

{

InitializeComponent();

using (StreamReader reader = new StreamReader("Name.txt"))

{

string text = reader.ReadToEnd();

lName.Text = text;

}

}

private void LoadData()

{

try

{

sqlDataAdapter = new SqlDataAdapter($"SELECT \*, N'Удалить' AS [Delete] FROM {table}", sqlConnection);

sqlCommandBuilder = new SqlCommandBuilder(sqlDataAdapter);

sqlCommandBuilder.GetInsertCommand();

sqlCommandBuilder.GetUpdateCommand();

sqlCommandBuilder.GetDeleteCommand();

dataSet = new DataSet();

sqlDataAdapter.Fill(dataSet, table);

dgvTransport.DataSource = dataSet.Tables[table];

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

{

DataGridViewLinkCell linkCell = new DataGridViewLinkCell();

dgvTransport[columnCount, i] = linkCell;

}

dgvTransport.Columns[0].HeaderText = "ID транспорта";

dgvTransport.Columns[1].HeaderText = "Номер транспорта";

dgvTransport.Columns[2].HeaderText = "Название";

dgvTransport.Columns[3].HeaderText = "Грузоподъемность, кг.";

dgvTransport.Columns[4].HeaderText = "ID водителя";

dgvTransport.Columns[5].HeaderText = "Функция";

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void ReloadData()

{

try

{

dataSet.Tables[table].Clear();

sqlDataAdapter.Fill(dataSet, table);

dgvTransport.DataSource = dataSet.Tables[table];

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

{

DataGridViewLinkCell linkCell = new DataGridViewLinkCell();

dgvTransport[columnCount, i] = linkCell;

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

this.Close();

}

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

{

ReloadData();

}

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

{

sqlConnection = new SqlConnection(@"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\User\Desktop\Курсач\YchetGryzoperevozok\YchetGryzoperevozok\Gryzoperevozki.mdf;Integrated Security=True");

sqlConnection.Open();

LoadData();

}

private void dgvTransport\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

try

{

if (e.ColumnIndex == columnCount)

{

string task = dgvTransport.Rows[e.RowIndex].Cells[columnCount].Value.ToString();

if (task == "Удалить")

{

if (MessageBox.Show("Удалить эту строку ?", "Удаление", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)

{

int rowIndex = e.RowIndex;

dgvTransport.Rows.RemoveAt(rowIndex);

dataSet.Tables[table].Rows[rowIndex].Delete();

sqlDataAdapter.Update(dataSet, table);

}

}

else if (task == "Добавить")

{

int rowIndex = dgvTransport.Rows.Count - 2;

DataRow row = dataSet.Tables[table].NewRow();

row["Nomer"] = dgvTransport.Rows[rowIndex].Cells["Nomer"].Value;

row["Name"] = dgvTransport.Rows[rowIndex].Cells["Name"].Value;

row["Сapacity"] = dgvTransport.Rows[rowIndex].Cells["Сapacity"].Value;

row["id driver"] = dgvTransport.Rows[rowIndex].Cells["id driver"].Value;

dataSet.Tables[table].Rows.Add(row);

dataSet.Tables[table].Rows.RemoveAt(dataSet.Tables[table].Rows.Count - 1);

dgvTransport.Rows.RemoveAt(dgvTransport.Rows.Count - 2);

dgvTransport.Rows[e.RowIndex].Cells[columnCount].Value = "Удалить";

sqlDataAdapter.Update(dataSet, table);

newRowAdding = false;

}

else if (task == "Обновить")

{

int r = e.RowIndex;

dataSet.Tables[table].Rows[r]["Nomer"] = dgvTransport.Rows[r].Cells["Nomer"].Value;

dataSet.Tables[table].Rows[r]["Name"] = dgvTransport.Rows[r].Cells["Name"].Value;

dataSet.Tables[table].Rows[r]["Сapacity"] = dgvTransport.Rows[r].Cells["Сapacity"].Value;

dataSet.Tables[table].Rows[r]["id driver"] = dgvTransport.Rows[r].Cells["id driver"].Value;

sqlDataAdapter.Update(dataSet, table);

dgvTransport.Rows[e.RowIndex].Cells[columnCount].Value = "Удалить";

}

ReloadData();

}

}

catch (SqlException ex)

{

MessageBox.Show("Значение уже используется в другой таблице, поэтому не может быть удалено или изменено!", "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void dgvTransport\_UserAddedRow(object sender, DataGridViewRowEventArgs e)

{

try

{

if (newRowAdding == false)

{

newRowAdding = true;

int lastRow = dgvTransport.Rows.Count - 2;

DataGridViewRow row = dgvTransport.Rows[lastRow];

DataGridViewLinkCell linkCell = new DataGridViewLinkCell();

dgvTransport[columnCount, lastRow] = linkCell;

row.Cells["Delete"].Value = "Добавить";

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void dgvTransport\_CellValueChanged(object sender, DataGridViewCellEventArgs e)

{

try

{

if (newRowAdding == false)

{

int rowIndex = dgvTransport.SelectedCells[0].RowIndex;

DataGridViewRow editingRow = dgvTransport.Rows[rowIndex];

DataGridViewLinkCell linkCell = new DataGridViewLinkCell();

dgvTransport[columnCount, rowIndex] = linkCell;

editingRow.Cells["Delete"].Value = "Обновить";

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void dgvTransport\_EditingControlShowing(object sender, DataGridViewEditingControlShowingEventArgs e)

{

e.Control.KeyPress -= new KeyPressEventHandler(Column\_KeyPress);

if (dgvTransport.CurrentCell.ColumnIndex == 3 || dgvTransport.CurrentCell.ColumnIndex == 4)

{

TextBox textBox = e.Control as TextBox;

if (textBox != null)

{

textBox.KeyPress += new KeyPressEventHandler(Column\_KeyPress);

}

}

}

private void Column\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar))

{

e.Handled = true;

}

}

}

}

# Приложение Б (Справочное) Выходные документы

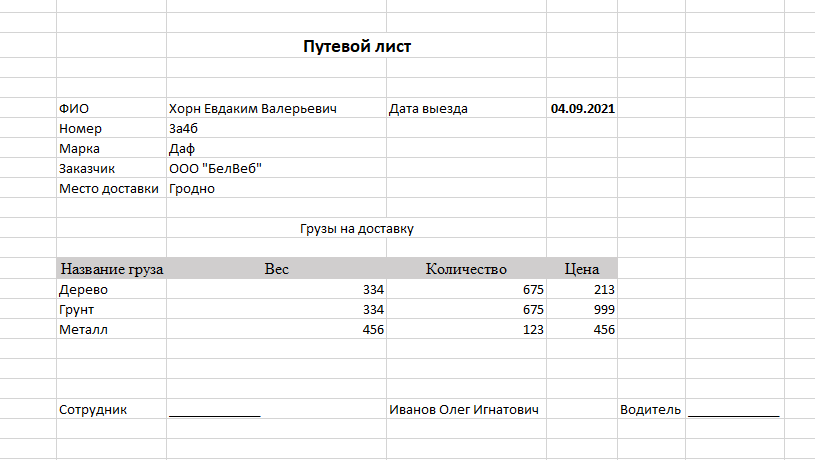


Рисунок Б.1 - Путевой лист



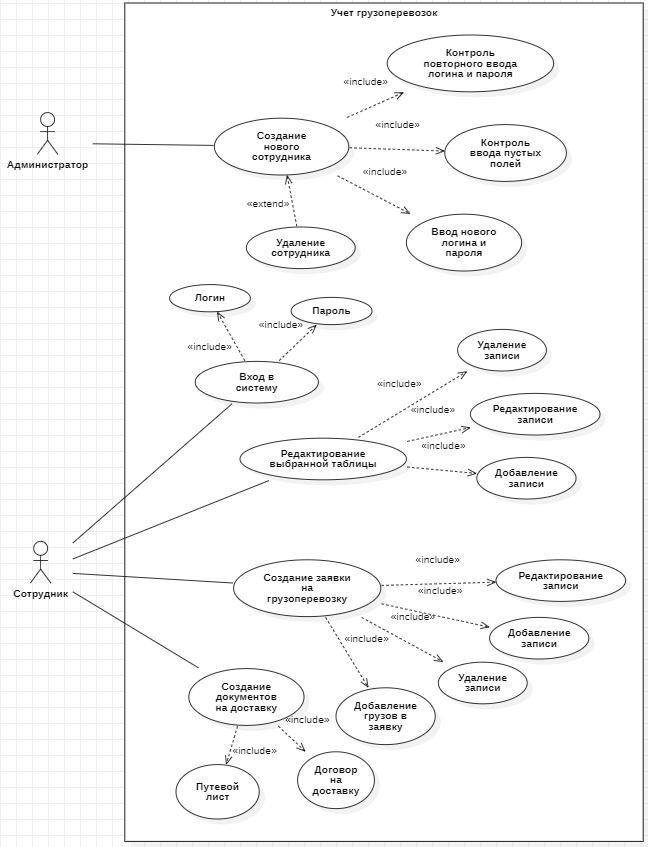
Рисунок Б.2 - Договор на перевозку груза

КП Т.992016.401 ПЗ

Подп. и дата

Инв.№ дубл.

Взам.инв.№



Подп. и дата

КП Т.992016.401 ПЗ

*Разработка программного средства для учета заявок на грузоперевозки*

Диаграмма вариантов использования

Масса

Лит.

Масштаб

№ докум.

Подпись

Дата

Изм.

Лист

*Миклашевич И.С.*

Разраб.

У

Багласова Е.В.

Провер.

Инв.№ подл.

Багласова Е.В.

Листов 5

Лист 1

Н.Контр.

КБП

Реценз.

Утверд.

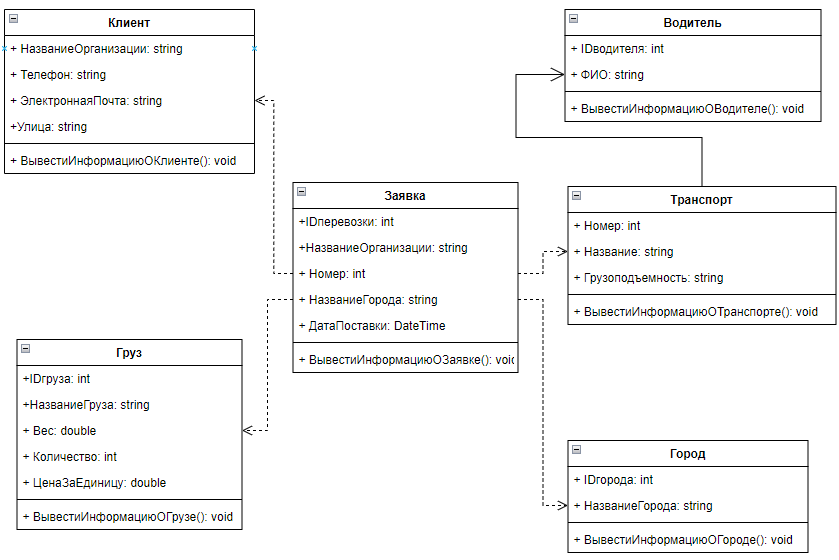
Т.Контр.

КП Т.992016.401 ПЗ

Подп. и дата

Инв.№ дубл.

Взам.инв.№



Подп. и дата

КП Т.992016.401 ПЗ

*Разработка программного средства для учета заявок на грузоперевозки*

Диаграмма классов

Масса

Лит.

Масштаб

№ докум.

Подпись

Дата

Изм.

Лист

*Миклашевич И.С.*

Разраб.

У

Багласова Е.В.

Провер.

Инв.№ подл.

Багласова Е.В.

Листов 5

Лист 2

Н.Контр.

КБП

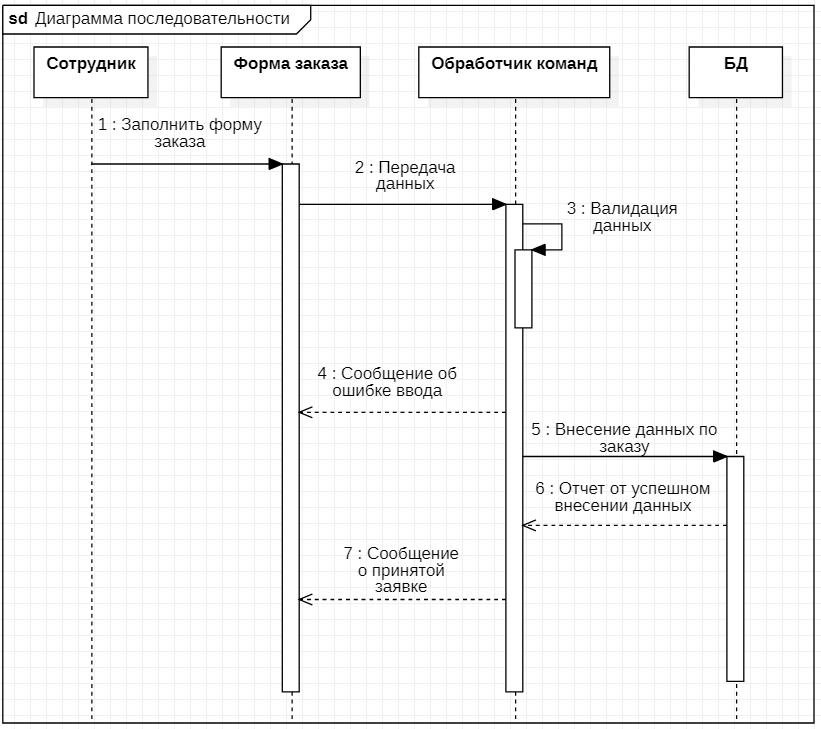
Реценз.

КП Т.992016.401 ПЗ

Подп. и дата

Инв.№ дубл.

Взам.инв.№



Подп. и дата

КП Т.992016.401 ПЗ

*Разработка программного средства для учета заявок на грузоперевозки*

Диаграмма последовательности

Масса

Лит.

Масштаб

№ докум.

Подпись

Дата

Изм.

Лист

*Миклашевич И.С.*

Разраб.

У

Багласова Е.В.

Провер.

Инв.№ подл.

Багласова Е.В.

Листов 5

Лист 3

Н.Контр.

КБП

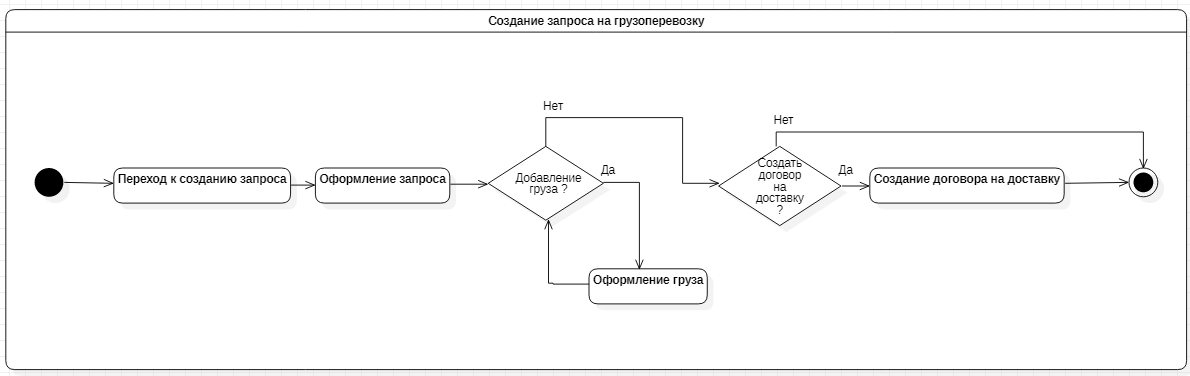
Реценз.

КП Т.992016.401 ПЗ

Подп. и дата

Инв.№ дубл.

Взам.инв.№



Подп. и дата

КП Т.992016.401 ПЗ

*Разработка программного средства для учета заявок на грузоперевозки*

Диаграмма состояния

Масса

Лит.

Масштаб

№ докум.

Подпись

Дата

Изм.

Лист

*Миклашевич И.С.*

Разраб.

У

Багласова Е.В.

Провер.

Инв.№ подл.

Багласова Е.В.

Листов 5

Лист 4

Н.Контр.

КБП

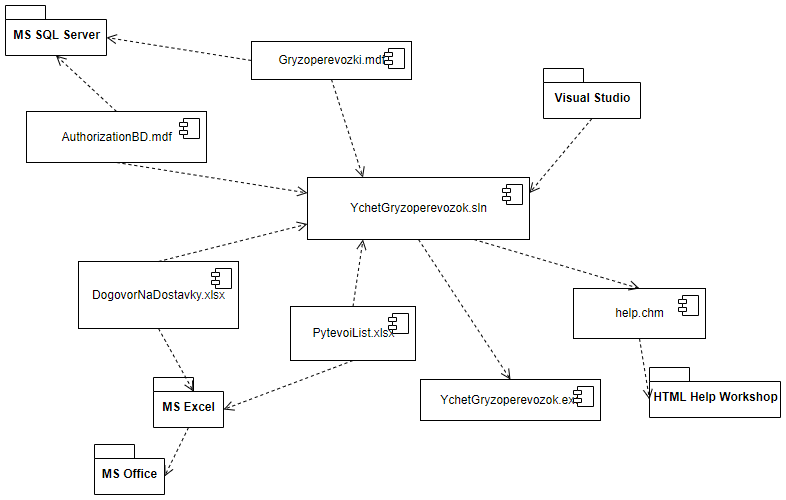
Реценз.

КП Т.992016.401 ПЗ

Подп. и дата

Инв.№ дубл.

Взам.инв.№



Подп. и дата

КП Т.992016.401 ПЗ

*Разработка программного средства для учета заявок на грузоперевозки*

Диаграмма компонентов

Масса

Лит.

Масштаб

№ докум.

Подпись

Дата

Изм.

Лист

*Миклашевич И.С.*

Разраб.

У

Багласова Е.В.

Провер.

Инв.№ подл.

Багласова Е.В.

Листов 5

Лист 5

Н.Контр.

КБП

Реценз.