### Частное учреждение образования

### «Колледж бизнеса и права»

### СОЗДАНИЕ ПРИЛОЖЕНИЯ ПО АВТОМАТИЗАЦИИ РАБОТЫ ТРАНСПОРТНО-ЛОГИСТИЧЕСКОГО ПРЕДПРИЯТИЯ «SUB-ZERO»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

### к курсовому проектированию по дисциплине

### «Конструирование программ и языки программирования»

КП Т.196012.401 ПЗ

Руководитель проекта (А.В.Кривошеина)

Учащийся (Д.В.Макатревич )

Минск, 2024

**СОДЕРЖАНИЕ**

Изм.

Изм.

Лист

Лист

№ докум.

№ докум.

Подпись

Подпись

Дата

Дата

Лист

Лист

3

3

КП Т.196012.401 ПЗ

ПЗ

ДП Т.517074.401 ПЗ

Разраб.

Разраб.

Макатревич Д.В.

Мовчан П.И.

Провер.

Провер.

Кривошеина А.В.

Багласова Е.В.

Т. контр.

Т. контр.

*Якимович К.О.*

Н. контр.

Н. контр.

Багласова Е.В.

Утверд.

Утверд.

*Багласова Т.Г.*

### *СОЗДАНИЕ ПРИЛОЖЕНИЯ ПО* *АВТОМАТИЗАЦИИ РАБОТЫ ТРАНСПОРТНО-ЛОГИСТИЧЕСКОГО ПРЕДПРИЯТИЯ «SUB-ZERO»*

*Программное средство для автоматизации учета заявок жителей Фрунзенского района г.Минска*

Лит.

Лит.

Листов

Листов

56

54

КБП

КБП

[Введение 4](#_Toc170137316)

[1. Описание задачи 5](#_Toc170137317)

[1.1. Анализ предметной области 5](#_Toc170137318)

[1.2. Постановка задачи 7](#_Toc170137319)

[2. Проектирование системы 8](#_Toc170137320)

[2.1. Требования к приложению 8](#_Toc170137321)

[2.2. Проектирование модели 8](#_Toc170137322)

[2.3. Организация данных 11](#_Toc170137323)

[2.4. Концептуальный прототип 13](#_Toc170137324)

[3. Описание реализации программного средства 19](#_Toc170137325)

[3.1. Инструменты разработки и применения технологии 19](#_Toc170137326)

[3.2. Порядок авторизации пользователей 20](#_Toc170137327)

[3.3. Организация данных 21](#_Toc170137328)

[3.4. Функции: логическая и физическая организация 24](#_Toc170137329)

[3.5. Входные и выходные данные 28](#_Toc170137330)

[3.6. Функциональное тестирование 29](#_Toc170137331)

[3.7. Описание справочной системы 34](#_Toc170137332)

[4. Применение 35](#_Toc170137333)

[4.1. Назначение программного средства 35](#_Toc170137334)

[4.2. Условия применения 35](#_Toc170137335)

[Заключение 37](#_Toc170137336)

[Список использованных источников 38](#_Toc170137337)

[ПРИЛОЖЕНИЕ А Текст программных модулей 40](#_Toc170137338)

[ПРИЛОЖЕНИЕ Б Формы выходных документов 46](#_Toc170137339)

[ПРИЛОЖЕНИЕ В Результаты работы программы 47](#_Toc170137340)

# **Введение**

В современном мире, где технологии проникают во все сферы жизни, отрасль транспортной логистики не является исключением. Автоматизация процессов становится все более актуальной и важной задачей, позволяющей минимизировать потери рабочего времени персонала, экономить на его численности и повышать уровень сервиса.

Целью курсового проекта «Автоматизированное рабочее место менеджера по работе с клиентами и сотрудника транспортного отдела в транспортно-логистической компании» является создание программного средства, которое позволит автоматизировать работу менеджера и сотрудника транспортного отдела в транспортно-логистической компании.

Пояснительная записка к курсовому проекту содержит пять разделов:

Описание задачи: В этом разделе раскрывается организационная сущность задачи, описывается предметная область и круг задач, которые должны быть решены. Описываются существующие аналоги.

Проектирование системы: В этом разделе перечисляются требования к аппаратному обеспечению и конфигурации компьютера, проводится характеристика операционной системы, обоснование выбранной среды для разработки приложения. Описываются требования к приложению, строится концептуальный прототип, описывается логическая и физическая организация данных, проектируется справочная система.

Описание реализации программного средства: В этом разделе представлены общие сведения о программном средстве и его функциональном назначении, входные и выходные данные.

Применение предназначен для описания сведений о назначении программного средства и области его применения. В этом разделе приводится структура справочной системы.

Заключение будет проанализировано созданное программное средство, определена степень соответствия поставленной задачи и выполненной работы.

Приложение А будет содержать текст программы.

Графическая часть содержит в себе все необходимые диаграммы для проектирования данного программного средства, такие как:

* диаграмма вариантов использования;
* диаграмма деятельности;
* диаграмма классов;
* диаграмма последовательности.

# **Описание задачи**

# **Анализ предметной области**

Логистика – наука об организации совместной деятельности менеджеров различных подразделений предприятия, группы предприятий в целях эффективного продвижения продукции по цепи «закупки сырья – производство продукции – сбыт – распределение», а также целенаправленных транспортно-людских потоков на основе интеграции и координации операций, процедур и функций, выполняемых в целях минимизации общих затрат ресурсов.

Транспортная логистика – это неотъемлемая часть современного бизнеса, которая играет ключевую роль в обеспечении эффективного движения товаров от производителя к конечному потребителю. Она включает в себя все аспекты управления цепочкой поставок, начиная от планирования и заканчивая доставкой товаров.

Важными элементами транспортной логистики являются:

* **управление запасами**: это включает в себя прогнозирование спроса, определение оптимального уровня запасов и управление процессом пополнения запасов;
* **управление транспортом**: это включает в себя выбор наиболее эффективных и экономически выгодных средств транспорта, планирование маршрутов и отслеживание движения грузов;
* **управление складом**: это включает в себя управление процессами приема, хранения и отпуска товаров на складе;
* **управление информацией**: это включает в себя сбор, обработку и анализ данных для поддержки принятия решений в области логистики;
* **управление поставками**: это включает в себя координацию и контроль над движением товаров от поставщика к потребителю. Это может включать в себя выбор поставщиков, управление заказами и контроль за выполнением контрактов;
* **управление складскими операциями**: это включает в себя управление процессами приема, хранения и отпуска товаров на складе. Это может включать в себя управление запасами, управление складским пространством и управление процессами погрузки и разгрузки;
* **управление транспортными операциями**: это включает в себя планирование и контроль над движением товаров от склада к потребителю. Это может включать в себя выбор транспортных средств, планирование маршрутов и управление процессами погрузки и разгрузки;
* **управление информацией**: это включает в себя сбор, обработку и распространение информации, необходимой для эффективного управления логистическими операциями. Это может включать в себя управление данными о товарах, поставщиках, заказах, доставках и другой информацией;
* **управление рисками**: это включает в себя идентификацию, оценку и управление рисками, связанными с логистическими операциями. Это может включать в себя риски, связанные с поставками, транспортировкой, складированием, информацией и другими аспектами логистики.

Внутренняя транспортная логистика занимается решением множества задач, которые контролируют транспортно-складские процессы. Особенности груза, такие как габариты и вес, напрямую влияют на выбор подходящей грузоподъёмности транспортного средства.

Логист выполняет следующе задачи:

* планирование маршрута;
* выбор оптимального способа доставки;
* производит расчёт стоимости грузоперевозки;
* сортировка, упаковка, привлечении грузчиков;
* хранение и консолидация грузов.

Промежуточные показатели, такие как статус доставки, пункт назначения, время в пути, могут быть получены через систему отслеживания на в приложении. Итоговые показатели, такие как общее время доставки, стоимость транспортировки, эффективность использования транспортных средств, могут быть рассчитаны на основе собранных данных.

Автоматизация процессов с помощью компьютерных систем позволяет значительно увеличить эффективность работы, минимизировать вероятность ошибок, связанных с человеческим фактором, и сократить время на выполнение рутинных задач.

Компьютерная обработка информации обеспечивает быстрый доступ к актуальным данным, возможность их анализа и принятия обоснованных решений на основе полученной информации. Это особенно важно в области транспортной логистики, где требуется оперативное реагирование на изменяющиеся условия и быстрое принятие решений.

Потенциальной аудиторией данного приложения являются транспортно-логистические компании, их сотрудники (логисты, менеджеры по работе с транспортом и водителями) и клиенты, которым необходима оперативная и точная информация о процессе доставки. Приложение также может быть полезно для водителей, которым необходимо координировать свои действия с центральным офисом. Кроме того, оно может быть полезно для поставщиков и производителей, которые хотят оптимизировать свои логистические операции и улучшить эффективность доставки своих товаров до конечного потребителя.

# **Постановка задачи**

Исходя из анализа предметной области, можно выделить следующие задачи, подлежащие автоматизации:

* автоматизировать отслеживание статуса доставки;
* автоматизировать управление запасами;
* автоматизировать выбор транспортного средства;
* автоматизировать создания груза;
* автоматизировать генерацию отчетов;

В сравнении с другими приложениями для транспортной логистики, такими как «1С:Предприятие», «SAP», «Relog», «Axelot», «Fretron», «Оracle Transportation Management» и другими, ваше приложение Sub-Zero имеет следующие преимущества:

* автоматическое управление запасами: Sub-Zero автоматически обновляет информацию о наличии товаров на основе данных о поставках и продажах, что помогает логистам определить, когда и сколько товаров нужно заказать, чтобы избежать избыточных запасов или дефицита товаров;
* гибкость и масштабируемость: Sub-Zero спроектирован так, чтобы быть гибким и масштабируемым, позволяя легко добавлять новые функции и поддерживать большое количество пользователей;
* пользовательский макет: Sub-Zero имеет интуитивно понятный и простой в использовании пользовательский Макет, что облегчит работу с ним для всех пользователей, независимо от их уровня технической грамотности.

Таким образом, Sub-Zero предлагает уникальный набор функций, которые помогут улучшить эффективность и производительность в области транспортной логистики, делая его важным инструментом для любого бизнеса, который хочет оставаться конкурентоспособным в современном быстро меняющемся бизнес-окружении.

# **Проектирование системы**

# **Требования к приложению**

В приложении Sub-Zero будет реализована система аутентификации для обеспечения безопасности данных пользователя. Это означает, что доступ к функционалу приложения будет возможен только после ввода уникальных учетных данных (имя пользователя и пароль). Это помогает предотвратить несанкционированный доступ и защитить конфиденциальность данных пользователя.

Макет приложения Sub-Zero будет разработан таким образом, чтобы обеспечить простоту и удобство использования. Однако, некоторые функции могут быть доступны только авторизованным пользователям. Кроме того, некоторые функции могут быть ограничены в зависимости от роли пользователя в системе.

Макет приложения Sub-Zero будет иметь четкую и последовательную структуру. Все элементы управления будут иметь стандартный размер, форму и расположение, чтобы облегчить навигацию пользователям. Шрифт будет выбран таким образом, чтобы обеспечить хорошую читаемость на всех устройствах. Кроме того, цветовая схема и стиль дизайна будут выбраны так, чтобы обеспечить приятное визуальное восприятие и удобство использования.

Основными минимальными требованиями, выдвигаемыми к аппаратному обеспечению персонального компьютера, являются:

* процессор 800 МГц и выше;
* оперативная память 128 Мбайт и более;
* свободное место на диске 100 Мбайт;
* интегрированная видеокарта на 512 Мбайт и более;
* монитор;
* мышь, клавиатура;
* принтер.

# **Проектирование модели**

Главной целью проектирования моделей является отображение функциональной структуры объекта, то есть производимые ими действия и связи между этими действиями.

Наиболее распространенные средства моделирования являются: диаграмма вариантов использования, диаграмма классов, диаграмма последовательности, диаграмма деятельности.

Диаграмма вариантов использования – это тип диаграммы, используемой в области разработки программного обеспечения для визуализации системы с точки зрения её актеров и вариантов использования.

Основные элементы диаграммы вариантов использования:

* актеры: – актеры представляют собой людей или системы, которые взаимодействуют с системой. Они находятся вне системы и могут быть как людьми (например, пользователи или администраторы), так и другими системами или устройствами;
* варианты использования: варианты использования – это функции или действия, которые система может выполнять в ответ на запросы от актеров. Они представляют собой цели, которые ­ актеры хотят достичь с помощью системы;
* отношения: отношения между актёрами и вариантами использования показывают, как – актеры и варианты использования взаимодействуют друг с другом. Они могут быть обозначены стрелками или линиями.

Диаграмма вариантов использования помогает разработчикам и стейкхолдерам лучше понять, как система будет работать и какие функции она будет выполнять. Она также помогает определить требования к системе и спланировать её разработку. Это важный инструмент в процессе проектирования системы.

Диаграмма вариантов использования представлена на листе 1 графической части.

На этой диаграмме можно выделить трёх пользователей программным продуктом. Клиент, который может обратиться к логисту с просьбой о перевозке товара. Логист, который добавляет клиента в систему и может его редактировать, а также создавать заявки на перевозку посылок. Сотрудник транспортного отдела, который создаёт новых водителе и транспорт.

Диаграмма классов – это структурная диаграмма языка моделирования UML, которая демонстрирует общую структуру иерархии классов системы, их атрибуты (полей), методы, Макеты и взаимосвязи (отношений) между ними. Она используется для визуализации структуры классов в системе и их взаимосвязей.

Вот основные элементы диаграммы классов:

* классы: представляют собой ключевые элементы в объектно-ориентированном моделировании. На диаграмме классы представлены в рамках, содержащих три компонента: имя класса, поля (атрибуты) класса и методы класса;
* отношения: между классами на диаграмме классов могут представлять отношения, такие как ассоциация, наследование и реализация.

Диаграмма классов соответствует принципам объектно-ориентированного программирования (ООП) и является одним из базовых инструментов проектирования ООП-систем. Она помогает лучше понимать структуру системы и ее компоненты.

В данном приложении присутствуют следующие классы:

* пользователь;
* посылка;
* город;
* машина;
* водитель;
* заказчик;
* доставка.

Диаграмма классов представлена на листе 2 графической части.

Диаграмма деятельности – это графическое представление процессов и задач, выполняемых в рамках определенного проекта или деятельности. Эта диаграмма используется для анализа и оптимизации бизнес-процессов, планирования работы, принятия решений и управления проектами.

Основные элементы диаграммы деятельности:

* начальный узел активности является первым элементом диаграммы деятельности и представляет начальную точку процесса;
* конечный узел активности является последним элементом диаграммы деятельности и обозначает завершение процесса;
* действия представляют собой отдельные шаги или операции, которые выполняются в процессе;
* поток управления показывают последовательность действий и передачу данных между действиями;
* узел принятия решений используются для представления ветвлений и слияний в потоке управления.

Диаграмма деятельности позволяет визуализировать последовательность задач и операций, которые нужно выполнить, чтобы достичь желаемого результата. Она является мощным инструментом для анализа, планирования и управления проектами и бизнес-процессами.

В рамках проекта рассмотрим функцию добавления заказа на перевозку.

Диаграмма деятельности для создания посылки представлена на листе 3 графической части.

Диаграмма последовательности – это UML-диаграмма, которая показывает взаимодействие между объектами в системе в виде последовательности сообщений, действий и операций. Она отображает порядок выполнения действий и обмена информацией между объектами во времени.

Вот основные элементы диаграммы последовательности:

* объекты представлены горизонтальной осью диаграммы. Каждый объект имеет свою колонку;
* сообщения передаются горизонтально между объектами и располагаются вертикально в хронологическом порядке. Они представлены стрелками;
* время представлено вертикальной осью диаграммы. Жизненный цикл объекта (создание-деятельность-уничтожение некой сущности) и взаимодействие актёров (действующих лиц) информационной системы в рамках прецедента показаны на единой временной оси.

Диаграмма последовательности помогает анализировать и понимать взаимодействие между объектами в системе, а также порядок, в котором происходит взаимодействие. Она является важным инструментом для анализа и проектирования систем.

Диаграмма последовательности для создания посылки представлена на листе 4 графической части.

# **Организация данных**

Наиболее распространенным средством моделирования таких типов документации являются схемы-сети (ERD), которые применяются для графического представления многих элементов разработанных программ систем и представляют некоторые габаритные размеры обозначений для определения данных и отношений между ними. С помощью такой диаграммы можно описать отдельные компоненты концептуальной модели или выразить свою точку зрения взаимосвязей между ними, иногда какое значение или реализацию системы. Основными потенциалом данной нотации является понятная сущность в каждом случае каждый рассматриваемый объект может явиться экземпляром одного и только одного сущности, должен иметь уникальное имя или идентификатор, а также отличается от других экземпляров данной сущности. Следовательно: как фотонные линии короткого ассортимента между отделами сущности.

Графическая модель данных строится таким образом, чтобы связь между отделениями сущности органа не только естественный характер соответствующего отношения, но и дополнительные аспекты обязательности связей.

Информационная модель представлена на диаграмме «Сущность-связь» на рисунки 2.1

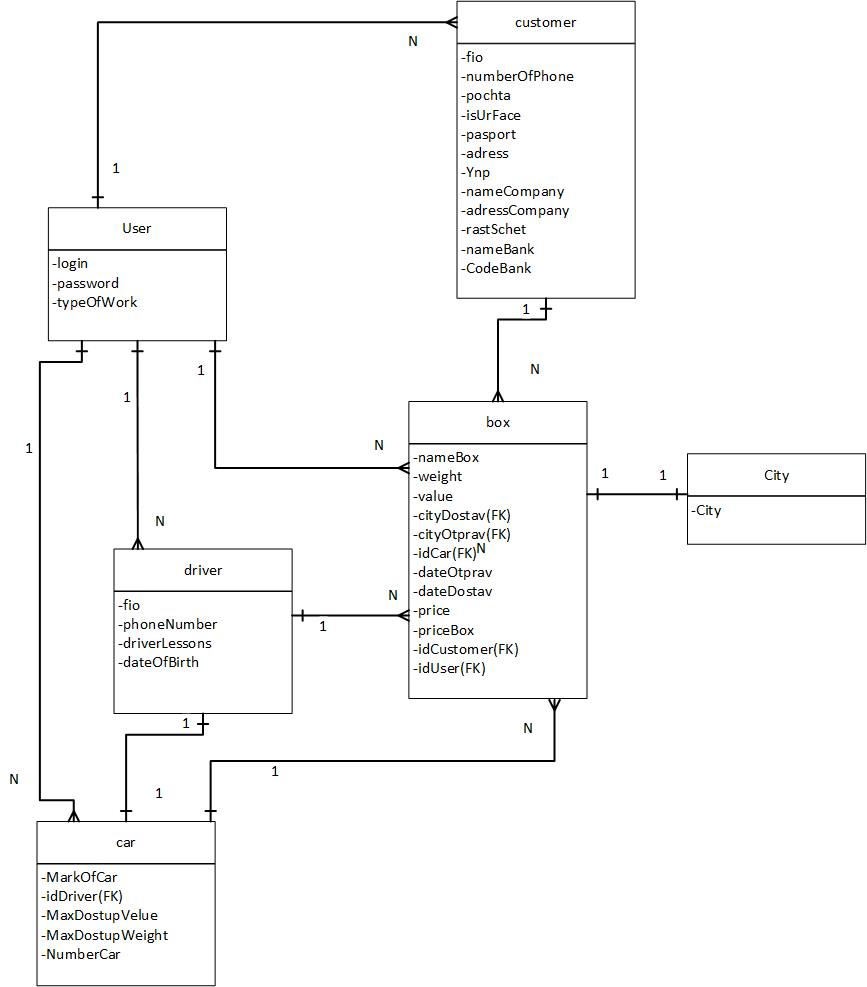


Рисунок 2.1 – Диаграмма «Сущность-связь»

Исследовав предметную область, можно выделить следующие сущности: «Посылка», «Город», «Водитель», «Машина», «Заказчик», «Пользователь».

Для сущности «Город» атрибутом будет название города.

Для сущности «Посылка» атрибутами будут являться:

* объём
* вес;
* город доставки;
* город отправки;
* цена посылки;
* название посылки;
* цена.

Для сущности «Машина» атрибутами будут являться:

* максимальный объём;
* максимальный вес;
* номер машины;
* марка машины;

Для сущности «Заказчик» атрибутами будут являться:

* ФИО;
* почта;
* адрес;
* УНП;
* код банка;
* название банка;
* расчетный счёт;
* номер работы
* адрес компании;
* название компании.

Для сущности «Водитель» атрибутами будут являться:

* ФИО;
* номер Телефона;
* водительские права;
* день рождения.

Для сущности «Пользователь» атрибутами будут являться:

* пароль;
* имя пользователя;
* тип работы;
* аватар.

Типы связей между сущностями:

* посылка-город: один ко одному;
* посылка-машина: один ко многим;
* машина-водитель: один ко одному;
* пользователь-посылка: один ко многим;
* заказчик-посылка: один ко одному.

# **Концептуальный прототип**

Концептуальный прототип состоит из описания внешнего вида пользовательского макета, а именно, элементов управления.

Разрабатываемое приложение будет содержать две основные формы: «Главная» и «Вход».

В макете формы «Входа» есть два поля для ввода текста, подписанные как «Логин» и «Пароль». Под полями для ввода находится кнопка с надписью «Войти», которую пользователи могут нажать для входа после ввода своих учетных данных. Внизу макета есть еще одна кнопка с надписью «Зарегистрироваться» для новых пользователей, которые хотят создать учетную запись.

Макет формы «Вход» представлен в приложении В на рисунке В.1.

В макете формы «Регистрация» есть два поля для ввода текста и поля для выбора деятельности, подписанные как «Логин» и «Пароль», и «Тип работы». Под полями для ввода находится кнопка с надписью «Зарегистрироваться», которую пользователи могут нажать для регистрации после ввода своих учетных данных

Макет формы «Регистрации» представлен в приложении В на рисунке В.2.

Макет главной формы для логистов состоит из:

* **поле «Поиск»**: Пользователи могут использовать это поле для поиска конкретной информации по посылкам;
* **кнопка «Добавить посылку»**: Эта кнопка позволяет пользователям добавлять новые посылки в систему;
* **таблица «Посылки»**: здесь отображается информация о всех посылках. Пользователи могут просматривать и управлять этой информацией;
* **кнопка «Распечатать документ»**: Пользователи могут использовать эту кнопку для печати документов, связанных с посылками и договорами;
* **кнопка «Заказчики»**: Эта кнопка позволяет пользователям просматривать информацию о заказчиках и управлять этой информацией;
* **кнопка «Выход»**: Пользователи могут использовать эту кнопку для выхода из системы.

Макет формы «Главной» представлен в приложении В на рисунке В.3.

Макет добавления посылки состоит из:

* **поле «Название посылки»**: Пользователь вводит название или идентификатор посылки;
* **поле «Заказчик»**: Пользователь выбирает заказчика из предложенных вариантов в выпадающем меню;
* **поле «Дата доставки»**: Пользователь выбирает или вводит желаемую дату доставки посылки;
* **поле «Вес посылки»**: Пользователь вводит вес посылки;
* **поле «Объём посылки»**: Пользователь вводит объем посылки;
* **поле «Водитель»**: Пользователь выбирает водителя из предложенных вариантов;
* **поле «Машина»**: Пользователь выбирает машину из предложенных вариантов;
* **поле «Город отправки»**: Пользователь выбирает город отправки из предложенных вариантов;
* **поле «Город получения»**: Пользователь выбирает город получения из предложенных вариантов;
* **поле «Цена посылки»**: Пользователь вводит стоимость доставки посылки;
* **кнопка «Создать»**: после заполнения всех полей пользователь нажимает эту кнопку для создания новой записи о посылке;
* **кнопка «Отмена»**: если пользователь решит не создавать новую запись, он может нажать эту кнопку, чтобы отменить действие и вернуться к предыдущему экрану.

Макет формы «Посылок» представлен в приложении В на рисунке В.4.

Макет формы «Заказчики» состоит из нескольких вариантов. Первый вариант – добавления физического лица в систему. Эта форма состоит из:

* **поле «ФИО»**: пользователь вводит полное имя физического лица;
* **поле «Номер тел.»**: пользователь вводит номер телефона физического лица;
* **поле «Почта»**: пользователь вводит адрес электронной почты физического лица;
* **поле «Паспорт»**: пользователь вводит паспортные данные физического лица;
* **поле «Адрес»**: пользователь вводит адрес проживания физического лица;
* **кнопка «Сохранить»**: после заполнения всех полей пользователь нажимает эту кнопку для сохранения информации о физическом лице;
* **кнопка «Отмена»**: если пользователь решит не сохранять введенную информацию, он может нажать эту кнопку, чтобы отменить действие и вернуться к предыдущему экрану.

Макет формы «Заказчики» представлен в приложении В на рисунке В.5.

Второй вариант – добавления юридической лица в систему. Эта форма состоит из:

* **поле «Номер тел.»**: Пользователь вводит номер телефона представителя юридического лица;
* **поле «Паспорт»**: Пользователь вводит паспортные данные представителя юридического лица;
* **поле «Адрес»**: Пользователь вводит адрес юридического лица;
* **поле «УНП»**: Пользователь вводит УНП (Учетный номер плательщика) юридического лица;
* **поле «Наименование компании»**: Пользователь вводит полное юридическое название компании;
* **поле «Адрес компании»**: Пользователь вводит юридический адрес компании;
* **поле «Название банка»**: Пользователь вводит название банка, в котором открыт счет компании;
* **поле «Расчет.счет»**: Пользователь вводит номер расчетного счета компании;
* **поле «Код банка»**: Пользователь вводит код банка;
* **кнопка «Сохранить»**: после заполнения всех полей пользователь нажимает эту кнопку для сохранения информации о юридическом лице;
* **кнопка «Отмена»**: если пользователь решит не сохранять введенную информацию, он может нажать эту кнопку, чтобы отменить действие и вернуться к предыдущему экрану.

Макет формы «Заказчики» представлен в приложении В на рисунке В.6.

Третий вариант – управление данными пользователя. Эта форма состоит из:

* **таблица «Пользователь»**: здесь отображается информация о пользователях. Пользователи могут просматривать и управлять этой информацией;
* **кнопка «Сохранить»**: после заполнения всех полей пользователь нажимает эту кнопку для сохранения информации;
* **кнопка «Отмена»**: если пользователь решит не сохранять введенную информацию, он может нажать эту кнопку, чтобы отменить действие и вернуться к предыдущему экрану.

Макет формы «Заказчики» представлен в приложении В на рисунке В.7.

Макет главной формы для сотрудника транспортного отдела состоит из:

* **поле «Поиск»**: Пользователь может использовать это поле для поиска конкретной информации в системе об автомобилях;
* **кнопка «Добавить машину»**: Эта кнопка позволяет пользователям добавлять новые машины в систему;
* **список «Водители»**: здесь отображается список водителей. Пользователи могут просматривать и управлять этой информацией;
* **таблица «Машины»**: здесь отображается информация о всех машинах. Пользователи могут просматривать и управлять этой информацией;
* **кнопка «Выход»**: Пользователи могут использовать эту кнопку для выхода из системы.

Макет формы «Главной» представлен в приложении В на рисунке В.8.

Макет формы «Водители» состоит из нескольких вариантов. Первый вариант – добавление водителя в систему. Эта форма состоит из:

* **поле «ФИО»**: пользователь вводит полное имя водителя;
* **поле «Номер телефона»**: пользователь вводит номер телефона водителя;
* **поле «Водительское уд»**: пользователь вводит номер водительского удостоверения;
* **поле «Дата рождения»**: пользователь вводит дату рождения водителя.
* **кнопка «Сохранить»**: после заполнения всех полей пользователь нажимает эту кнопку для сохранения информации о водителе;

**Кнопка «Отмена»**: если пользователь решит не сохранять введенную информацию, он может нажать эту кнопку, чтобы отменить действие и вернуться к предыдущему экрану.

Макет формы «Водители» представлен в приложении В на рисунке В.9.

Второй вариант – управление данными водителей. Эта форма состоит из:

* **таблица «Водители»**: здесь отображается информация о водителях. Пользователи могут просматривать и управлять этой информацией;
* **кнопка «Сохранить»**: после заполнения всех полей пользователь нажимает эту кнопку для сохранения информации;
* **кнопка «Отмена»**: если пользователь решит не сохранять введенную информацию, он может нажать эту кнопку, чтобы отменить действие и вернуться к предыдущему экрану.

Макет формы «Водители» представлен в приложении В на рисунке В.10.

Макет добавления машины состоит из:

* **поле «Марка автомобиля»**: Пользователь вводит марку автомобиля;
* **поле «Водитель»**: Пользователь выбирает водителя из предложенных вариантов;
* **поле «Макс допустимый объем»**: пользователь вводит максимально допустимый объем груза для автомобиля;
* **поле «Макс допустимый вес»**: пользователь вводит максимально допустимый вес груза для автомобиля;
* **поле «Номер машины»**: пользователь вводит номер автомобиля;
* **кнопка «Сохранить»**: после заполнения всех полей пользователь нажимает эту кнопку для сохранения информации об автомобиле;
* **кнопка «Отмена»**: если пользователь решит не сохранять введенную информацию, он может нажать эту кнопку, чтобы отменить действие и вернуться к предыдущему экрану.

Макет формы «Машины» представлен в приложении В на рисунке В.11.

Макет создания документа состоит из:

* **поле «Вид документа»**: пользователь выбирает из предложенных типов документов, где хочет создать отчет;
* **список «Пользователь»**: пользователь выбирает заказчиков из предложенных вариантов;
* **поле «Дата с\ дата по «**: пользователь вводит промежуток времени для создания отчета;
* **кнопка «Создать»**: после заполнения всех полей пользователь нажимает эту кнопку для документа об посылках пользователя;
* **кнопка «Отмена»**: если пользователь решит не сохранять введенную информацию, он может нажать эту кнопку, чтобы отменить действие и вернуться к предыдущему экрану.

Макет формы «Документы» представлен в приложении В на рисунке В.12.

Макет отметки доставки состоит из:

* **список «Название посылки»**: Пользователь выбирает посылку из предложенных вариантов;
* **список «Тип доставки»**: Пользователь выбирает тип доставки (забран со склада, храниться на складе);
* **поле «Дата получения»**: Пользователь вводит дату, когда посылку забрали, лежит на складе;
* **кнопка «Сохранить»**: после заполнения всех полей пользователь нажимает эту кнопку для сохранения информации об посылке.
* **кнопка «Отмена»**: если пользователь решит не сохранять введенную информацию, он может нажать эту кнопку, чтобы отменить действие и вернуться к предыдущему экрану.

Макет формы «Отметки» представлен в приложении В на рисунке В.13.

# **Описание реализации программного средства**

# **Инструменты разработки и применения технологии**

Инструментами разработки для будущего программного приложения будут являться:

* операционная система MS Widows 11;
* программная среда разработки Microsoft Visual Studio 2022;
* язык программирования C#;
* система управления реляционными базами данных (СУБД) Microsoft SQL Server 2024;
* программа для построения блок-схем Microsoft Visio
* язык определения данных DDL;
* справочная система Dr.Explain;
* офисный пакет приложений Microsoft Office.

MS Windows 11 – операционная система для персональных компьютеров и рабочих станций, разработанная корпорацией Microsoft в рамках семейства MS Windows. Доступна единая платформа разработки и единый магазин универсальных приложений, совместимых со всеми поддерживаемыми устройствами [19].

Microsoft Visual Studio 2022 – это набор инструментов для создания программного обеспечения: от планирования до разработки пользовательского Макета, написания кода, тестирования, отладки, анализа качества кода и производительности, развертывания в средах клиентов и сбора данных телеметрии по использованию [14].

MS Visual Studio 2022 включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и отладчик машинного уровня [14].

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 [15].

DDL состоит из набора SQL-команд, используемых для создания, изменения и удаления структур баз данных, но не данных. Он просто имеет дело с описаниями схемы базы данных и используется для создания и изменения структуры объектов, присутствующих в базе данных. Все DDL-команды автоматически фиксируются, что означает, что все изменения навсегда сохраняются в базе данных. DDL подразделяется на пять команд, которые широко используются в SQL-запросах[15].

Microsoft Visio – векторный графический редактор, редактор диаграмм и блок-схем для Windows. Выпускается в трёх редакциях: Standard, Professional и Pro for Office 365.Аналогично с Adobe Reader, в стандартный набор программ Microsoft Office входит только средство для просмотра и печати диаграмм Microsoft Visio Viewer. Полнофункциональная версия Microsoft Visio Professional для создания и редактирования монограмм и диаграмм. Первоначально Visio разрабатывался и выпускался компанией Shapeware, затем переименованной в Visio Corporation. Microsoft приобрела компанию в 2000 году, тогда продукт назывался Visio 2000 [18].

C# – объектно-ориентированный язык программирования общего назначения. Разработан в 1998—2001 годах группой инженеров компании Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота[6] как язык разработки приложений для платформы Microsoft .NET Framework и .NET Core. C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, переменные, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML [16].

Dr.Explain – это профессиональная программа для создания электронных справочных руководств, пользовательских инструкций и баз знаний. Она автоматизирует создание скриншотов, аннотацию изображений, индексацию контента и форматирование документации.

Microsoft Office – это офисный пакет, разработанный американской корпорацией Microsoft. В его состав входят приложения для работы с текстовыми документами, электронными таблицами, базами данных и другими типами файлов. Он предоставляет инструменты, такие как Word, Excel, PowerPoint, Outlook, Teams и многое другое [17].

# **Порядок авторизации пользователей**

В приложении можно выделить следующие роли и функции:

Сотрудник транспортного отдела:

* аутентификация: Сотрудник транспортного отдела регистрируется в системе, предоставляя необходимые учетные данные (например, имя пользователя и пароль);
* авторизация: после успешной аутентификации сотрудник транспортного отдела получает доступ к функциям, связанным с его ролью;
* функции: Создание профилей водителей и машин. Это может включать ввод информации о и информации о машине.

Логист:

* аутентификация: Логист регистрируется в системе, предоставляя необходимые учетные данные;
* авторизация: после успешной аутентификации логист получает доступ к функциям, связанным с его ролью;
* функции: Создание профилей клиентов и загрузка посылок. Это может включать ввод информации о клиенте (и информации о посылке.

# **Организация данных**

Организация данных подразумевает создание модели данных, главными элементами, которые являются сущности и их связи.

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 их атрибутами.

Структура базы данных разрабатываемого программного средства включает шесть таблиц.

Таблица «box» содержит информацию о посылках на перевозку, структура таблицы представлена в таблице 3.1

Таблица 3.1 – box

|  |  |  |  |
| --- | --- | --- | --- |
| Название строки | Тип данных | Размер в байтах | Описание |
| id | int | 4 | Номер посылки |
| nameBox | varchar | 250 | Название посылки |
| weight | float | 4 | Вес |
| value | float | 8 | Объём |
| cityDostav | int | 4 | Город доставки |
| cityOtprav | int | 4 | Город отправки |
| idCar | int | 4 | Номер машины |
| dateOtprav | datetime | 3 | Дата отправки |
| datePolych | datetime | 3 | Дата получения |
| price | float | 8 | Цена груза |

Продолжение таблицы 3.1

|  |  |  |  |
| --- | --- | --- | --- |
| Название строки | Тип данных | Размер в байтах | Описание |
| priceBox | float | 8 | Цена перевозки |
| idCustomer | int | 4 | Номер заказчика |
| idUser | int | 4 | Номер работника |

Таблица «Car» содержит информацию о автомобилях организации, структура таблицы представлена в таблице 3.2

Таблица 3.2 – Car

|  |  |  |  |
| --- | --- | --- | --- |
| Название строки | Тип данных | Размер в байтах | Описание |
| id | int | 4 | Номер машины |
| MarkOfCar | varchar | 250 | Марка машины |
| idDriver | int | 4 | Номер водителя |
| MaxDostupVelue | float | 8 | Максимальный объём |
| MaxDostupWeight | float | 8 | Максимальный вес |
| NumberCar | varchar(250) | 250 | Номер автомобиля |

Таблица «City» содержит информацию о городах доставки, структура таблицы представлена в таблице 3.3

Таблица 3.3 – City

|  |  |  |  |
| --- | --- | --- | --- |
| Название строки | Тип данных | Размер в байтах | Описание |
| id | int | 4 | Номер города |
| City | varchar(250) | 250 | Название города |

Таблица «Customer» содержит информацию о заказчиках, структура таблицы представлена в таблице 3.4

Таблица 3.4 – Customer

|  |  |  |  |
| --- | --- | --- | --- |
| Название строки | Тип данных | Размер в байтах | Описание |
| id | int | 4 | Номер заказчика |
| fio | varchar | 250 | ФИО |
| numberOfPhone | varchar | 250 | Номер телефона |
| pochta | varchar | 250 | Почта |
| isUrFace | int | 4 | Юр. |
| pasport | varchar | 250 | паспорт |
| adress | varchar | 250 | Адрес |
| ynp | varchar | 250 | УНП |
| nameCompany | varchar | 250 | Название компании |
| adressCompany | varchar | 250 | Адрес компании |
| rastSchet | Varchar | 250 | Расчетный счет |

Продолжение таблицы 3.4

|  |  |  |  |
| --- | --- | --- | --- |
| Название строки | Тип данных | Размер в байтах | Описание |
| nameBank | Varchar | 250 | Название банка |
| codeBank | Varchar | 250 | Код банка |

Таблица «driver» содержит информацию о водителях, структура таблицы представлена в таблице 3.5

Таблица 3.5 – driver

|  |  |  |  |
| --- | --- | --- | --- |
| Название строки | Тип данных | Размер в байтах | Описание |
| id | int | 4 | Номер водителя |
| fio | varchar | 250 | ФИО |
| numberOfPhone | varchar | 250 | Номер телефона |
| driverLessons | varchar | 250 | Номер прав |
| dateOfBirth | Datetime | 3 | День рождения |

Таблица «user» содержит информацию о пользователях, структура таблицы представлена в таблице 3.6

Таблица 3.6 – user

|  |  |  |  |
| --- | --- | --- | --- |
| Название строки | Тип данных | Размер в байтах | Описание |
| id | int | 4 | Номер пользователя |
| nickname | varchar | 250 | Логин |
| password | varchar | 250 | Пароль |
| kindOfWork | varchar | 250 | Тип работы |
| avatar | varchar | 250 | Путь к аватару |

Все таблицы представлены на рисунке 3.1

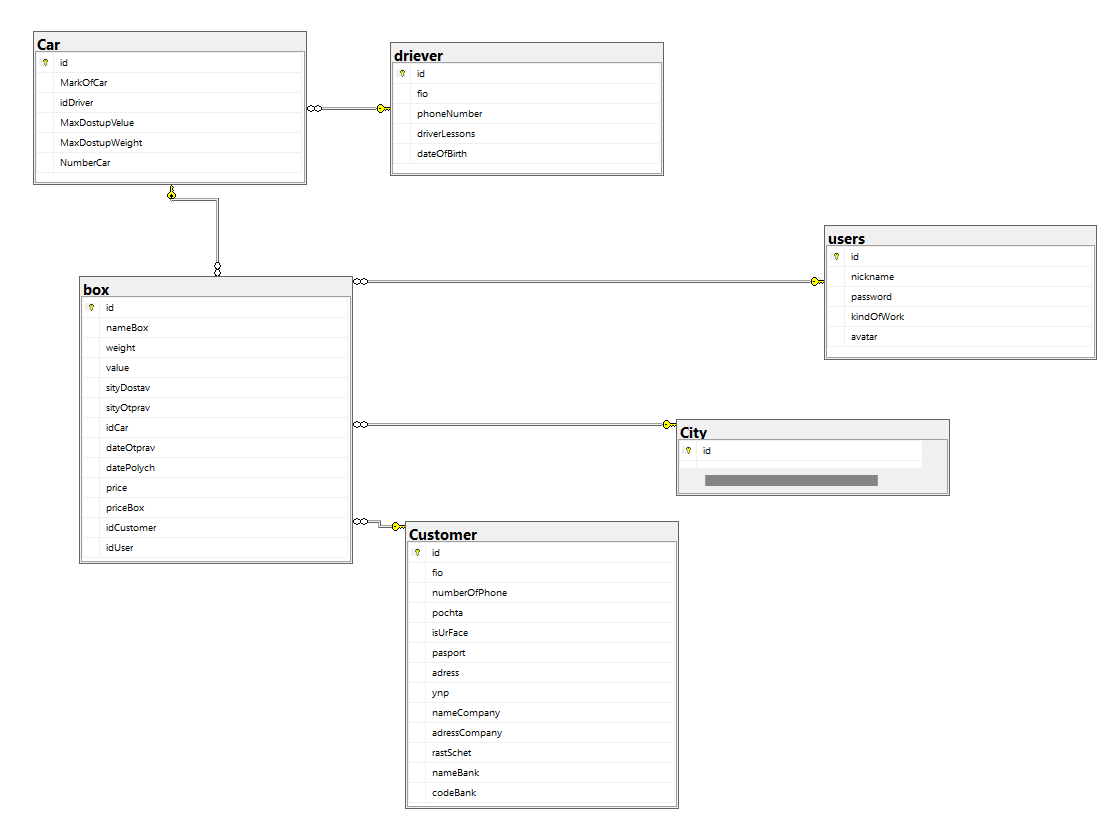


Рисунок 3.1 – Схема данных

# **Функции: логическая и физическая организация**

В данном проекте реализованы функции добавления, редактирования и удаления информации о посылке, об автомобилях, об заказчиках, об водителях; формирование договора на заключение грузоперевозки и информация об посылках заказчика; формирование поиска.

Функция добавления информации о посылке находиться на форме «Добавление посылки». После заполнения всех полей и нажатия на кнопку «Добавить» на данной форме происходит обработка события Button\_Click\_AddBox. Все данные, вводимые в поля формы, будут сохранены в таблице «Box». Код функции добавления посылки:

public static bool AddBox(Box box, MyUsers s)

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string carQuery = "SELECT \* FROM Car WHERE id = @IdCar ";

using (SqlCommand carCommand = new SqlCommand(carQuery, connection))

{

carCommand.Parameters.AddWithValue("@IdCar", box.Car.id);

using (SqlDataReader carReader = carCommand.ExecuteReader())

{

if (carReader.Read())

{

double maxDostupWeight = Convert.ToDouble(carReader["MaxDostupWeight"].ToString());

double maxDostupVelue = Convert.ToDouble(carReader["MaxDostupVelue"].ToString());

double Weight = string.IsNullOrEmpty(carReader["Weight"].ToString()) ? 0 : Convert.ToDouble(carReader["Weight"].ToString());

double Velue = string.IsNullOrEmpty(carReader["Velue"].ToString()) ? 0 : Convert.ToDouble(carReader["Velue"].ToString());

if (box.Weight > maxDostupWeight || box.Value > maxDostupVelue || maxDostupWeight < Weight + box.Weight || maxDostupVelue < Velue + box.Value)

{

return false;

}

}

else

{

return false;

}

}

}

int idBox;

string query = "INSERT INTO box (nameBox, weight, value, sityDostav, sityOtprav, idCar, dateOtprav, datePolych, price, priceBox, idCustomer,idUser) OUTPUT INSERTED.id VALUES (@NameBox, @Weight, @Value, @SityDostav, @SityOtprav, @IdCar, @dateOtprav, @datePolych, @Price, @PriceBox, @IdCustomer,@IdUser)";

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@NameBox", box.NameBox);

command.Parameters.AddWithValue("@Weight", box.Weight);

command.Parameters.AddWithValue("@Value", box.Value);

command.Parameters.AddWithValue("@SityDostav", box.SityDostav);

command.Parameters.AddWithValue("@SityOtprav", box.SityOtprav);

command.Parameters.AddWithValue("@IdCar", box.Car.id);

command.Parameters.AddWithValue("@dateOtprav", box.dateOtprav);

command.Parameters.AddWithValue("@datePolych", box.datePolych);

command.Parameters.AddWithValue("@Price", box.Price);

command.Parameters.AddWithValue("@PriceBox", box.PriceBox);

command.Parameters.AddWithValue("@IdCustomer", box.Customer.Id);

command.Parameters.AddWithValue("@IdUser", s.id);

idBox = (int)command.ExecuteScalar();

}

string insertDostavkaQuery = "INSERT INTO Dostavka ( DateDostav,idBox) VALUES (@DateDostav, @IdBox)";

using (SqlCommand insertDostavkaCommand = new SqlCommand(insertDostavkaQuery, connection))

{

insertDostavkaCommand.Parameters.AddWithValue("@IdBox", idBox);

insertDostavkaCommand.Parameters.AddWithValue("@DateDostav", box.datePolych);

insertDostavkaCommand.ExecuteNonQuery();

}

string updateCarQuery = "UPDATE Car SET Weight = Weight + @Weight, Velue = Velue + @Value WHERE id = @IdCar";

using (SqlCommand updateCarCommand = new SqlCommand(updateCarQuery, connection))

{

updateCarCommand.Parameters.AddWithValue("@Weight", box.Weight);

updateCarCommand.Parameters.AddWithValue("@Value", box.Value);

updateCarCommand.Parameters.AddWithValue("@IdCar", box.Car.id);

updateCarCommand.ExecuteNonQuery();

}

}

return true;

}

Функция редактирования информации о посылке находится на форме «Добавление посылки». После заполнения всех полей и нажатия на кнопку «Сохранить» на данной форме происходит обработка событий Button\_Click.Все данные, вводимые в поля формы, будут сохранены в таблице «Box».

Функция удаления информации о посылке находится на форме «Главная». После нажатия на кнопку «Удалить» на данной форме происходит обработка событий Button\_Click\_delet. Выбранные данные будут удалены из таблицы «Box».

Функция добавления информации об автомобилях находиться на форме «Автомобили». После заполнения всех полей и нажатия на кнопку «Добавить» на данной форме происходит обработка события Button\_Click. Все данные, вводимые в поля формы, будут сохранены в таблице «car».

Функция редактирования информации об автомобилях находится на форме «Добавление машины». После заполнения всех полей и нажатия на кнопку «Сохранить» на данной форме происходит обработка событий Button\_Click.Все данные, вводимые в поля формы, будут сохранены в таблице «car».

Функция удаления информации об автомобилях находится на форме «Главная». После нажатия на кнопку «Удалить» на данной форме происходит обработка событий Button\_Click\_delet. Выбранные данные будут удалены из таблицы «car».

Функция добавления информации об заказчиках находиться на форме «Добавление заказчика». После заполнения всех полей и нажатия на кнопку «Добавить» на данной форме происходит обработка события Button\_Click\_addPeople. Все данные, вводимые в поля формы, будут сохранены в таблице «Customer».

Функция редактирования информации об заказчиках находится на форме «Добавление заказчика». После заполнения всех полей и нажатия на кнопку «Сохранить» на данной форме происходит обработка событий Button\_Click\_addPeople. Все данные, вводимые в поля формы, будут сохранены в таблице «Customer».

Функция удаления информации об заказчиках находится на форме «Главная». После нажатия на кнопку «Удалить» на данной форме происходит обработка событий Button\_Click\_delet. Выбранные данные будут удалены из таблицы «Customer».

Функция добавления информации об водителях находиться на форме «Водители». После заполнения всех полей и нажатия на кнопку «Добавить» на данной форме происходит обработка события Button\_Click\_addPeople. Все данные, вводимые в поля формы, будут сохранены в таблице «driever».

Функция редактирования информации об водителях находится на форме «Водители». После заполнения всех полей и нажатия на кнопку «Сохранить» на данной форме происходит обработка событий Button\_Click.Все данные, вводимые в поля формы, будут сохранены в таблице «driever».

Функция удаления информации об водителях находится на форме «Водители». После нажатия на кнопку «Удалить» на данной форме происходит обработка событий Button\_Click\_delet. Выбранные данные будут удалены из таблицы «driever».

Функция формирования созданного договора происходит при нажатии на кнопку «Создать» на форме «Документы». Происходит обработка события Button\_Click\_1, в процессе которого будет создан документ Word со всеми данными в договоре. Код функции создания документа word:

public static void GenerateWordReport(Customer selectedCustomer, Box box, Drivers driver)

{

using (WordprocessingDocument wordDocument = WordprocessingDocument.Create("C:\\Users\\kupit\\OneDrive\\Desktop\\Отчет.docx", WordprocessingDocumentType.Document))

{

MainDocumentPart mainPart = wordDocument.AddMainDocumentPart();

mainPart.Document = new Document();

Body body = mainPart.Document.AppendChild(new Body());

AppendParagraph(body, "ДОГОВОР ПЕРЕВОЗКИ ГРУЗА", true, JustificationValues.Center);

AppendParagraph(body, $"{DateTime.Now:dd.MM.yyyy}.", false, JustificationValues.Right);

AppendParagraph(body, "Наименование компании: Sub\_Zero", false, JustificationValues.Right);

AppendParagraph(body, "1. СТОРОНЫ ДОГОВОРА", true);

AppendParagraph(body, $"1.1. Перевозчик: {driver.FullName}");

AppendParagraph(body, $"1.2. Грузополучатель: {selectedCustomer.Fio}");

AppendParagraph(body, $"1.3. Грузоотправитель: {selectedCustomer.Fio}");

AppendParagraph(body, "2. ХАРАКТЕРИСТИКИ ГРУЗА", true);

AppendParagraph(body, $"2.1. Вес: {box.Weight}");

AppendParagraph(body, $"2.2. Объем: {box.Value}");

AppendParagraph(body, "3. ПУНКТ ОТПРАВЛЕНИЯ И НАЗНАЧЕНИЯ", true);

AppendParagraph(body, $"3.1. Пункт отправления: {GetCityIdByName(box.SityOtprav)}");

AppendParagraph(body, $"3.2. Пункт назначения: {GetCityIdByName(box.SityDostav)}");

AppendParagraph(body, "4. СТОИМОСТЬ ДОСТАВКИ И ПОРЯДОК РАСЧЕТОВ", true);

AppendParagraph(body, $"4.1. Стоимость доставки: {box.PriceBox – box.Price}");

AppendParagraph(body, "5. СРОКИ ПЕРЕВОЗКИ", true);

AppendParagraph(body, $"5.1. Дата отправки: {box.dateOtprav}");

AppendParagraph(body, $"5.2. Дата получения: {box.datePolych}");

AppendParagraph(body, "6. ПРАВА И ОБЯЗАННОСТИ СТОРОН", true);

AppendParagraph(body, "6.1. Перевозчик обязуется доставить груз в указанное место в указанные сроки. Грузоотправитель обязуется оплатить услуги перевозки в соответствии с договором. Перевозчик не несет ответственности за потерю или повреждение груза.");

AppendParagraph(body, "7. ОТВЕТСТВЕННОСТЬ ЗА НАРУШЕНИЕ УСЛОВИЙ СДЕЛКИ", true);

AppendParagraph(body, "7.1. В случае нарушения условий договора стороны несут ответственность в соответствии с действующим законодательством.");

AppendParagraph(body, "8. ФОРС-МАЖОР", true);

AppendParagraph(body, "8.1. В случае наступления обстоятельств непреодолимой силы, стороны освобождаются от ответственности за неисполнение обязательств по договору.");

if (selectedCustomer.IsUrFace == 1)

{

AppendParagraph(body, "9. ЮРИДИЧЕСКИЕ АДРЕСА И БАНКОВСКИЕ РЕКВИЗИТЫ СТОРОН", true);

AppendParagraph(body, $"9.1. Юридический адрес: {selectedCustomer.AdressCompany}");

AppendParagraph(body, $"9.2. Банковские реквизиты: Банк – {selectedCustomer.NameBank}, Код банка – {selectedCustomer.CodeBank}");

}

AppendParagraph(body, "10. ПОДПИСИ СТОРОН", true);

AppendParagraph(body, "Грузоотправитель (Подпись)\_\_\_\_\_\_\_\_\_\_\_\_\_");

AppendParagraph(body, "Директор (Подпись)\_\_\_\_\_\_\_\_\_\_\_\_\_");

mainPart.Document.Save();

}

}

Функция формирования созданного документа информации об посылках заказчика происходит при нажатии на кнопку «Создать» на форме «Документы». Происходит обработка события Button\_Click\_1, в процессе которого будет создан документ Excel со всеми данными о посылках.

Полный код программных модулей представлен в приложении А.

# **Входные и выходные данные**

Входными данными являются данные, заносимые программой в таблицы: «Car», «driever», «Customer», «box». Также входными данными являются данные, вводимые в поля для поиска.

Для добавления данных в таблицу «Car» необходимо ввести следующие данные:

* марка машины;
* водитель;
* максимальный допустимый вес;
* максимальный допустимый размер;
* номер машины.

Для добавления данных в таблицу «driever» необходимо ввести следующие данные:

* ФИО;
* номер телефона;
* номер водительских прав;
* день рожденья.

Для добавления данных в таблицу «Customer» необходимо ввести следующие данные:

* ФИО;
* номер телефона;
* почта;
* паспорт;
* адрес;
* УНП;
* название компании;
* адрес компании;
* расчетный счет;
* название банка;
* код банка.

Для добавления данных в таблицу «box» необходимо ввести следующие данные:

* название посылки;
* вес;
* размер;
* машина;
* дата отправки;
* дата получения;
* цена;
* цена посылки;
* город отправки;
* город доставки.

Выходные данными являются данные, которые экспортируются в файлы Microsoft Office, а именно договор поставки. А также отчёт:

* отчёт по поставкам – документ представлен в приложении Б на рисунке Б.1;
* отчёт о посылках пользователя – документ представлен в приложении Б на рисунке Б.2.

Договор генерируется по нажатию на кнопку «Создать».

# **Функциональное тестирование**

Функциональное тестирование – это тестирование функций приложения на соответствия требованиям. Оценка производится в соответствии с ожидаемым и фактическим результатом.

Тестирование программы будет производится последовательно, переходя из одной программы в другую. Во время теста будут проверятся все действия с программой, навигация пунктов меню, которые может произвести пользователь. После чего, все собранные и найденные ошибки будут исправлены.

В таблицах 3.7-3.14 представлены тест-кейсы, подготовленные для проведения функционального тестирования.

Первый тест будет проведён в окне авторизации программы.

Тест-кейс для тестирования функции авторизации представлен в таблице 3.7

Таблица 3.7 – Тест-кейс функции авторизации

|  |  |  |
| --- | --- | --- |
| Функция | Шаги выполнения | Результат |
| Авторизация в приложении | 1.Заполнить поля данных:  – логин: «Doshik»;  – пароль: «1212».  2. Нажать на кнопку «Вход» | Ожидаемый результат: переход на форму «Главная» |
| Фактический результат: успешный переход на форму «Главная». Результат представлен в приложении В на рисунке В.14. |

Второй тест будет проведён на форме «Начальная».

Тест-кейс для тестирования функции регистрации пользователя будет представлен в таблицы 3.8

Таблица 3.8 – Тест-кейс функции регистрации

|  |  |  |
| --- | --- | --- |
| Функция | Шаги выполнения | Результат |
| Регистрация в приложении | 1.Заполнить поля данных:  – логин: «Polina»;  – тип деятельности: «Сотрудник транспортного отдела»;  – пароль: «1111».  2. Нажать на кнопку «Вход» | Ожидаемый результат: переход на форму «Главная» |
| Фактический результат: успешный переход на форму «Главная». Результат представлен в приложении В на рисунке В.15. |

Третий тест будет проведён на форме «Заказчики».

Тест-кейс для тестирования функции добавления заказчика будет представлен в таблицы 3.9

Таблица 3.9 – Тест-кейс функции добавления заказчика

|  |  |  |
| --- | --- | --- |
| Функция | Шаги выполнения | Результат |
| Добавления заказчика | 1.Заполнить поля данных:  - тип заказчика: «Физ.Лицо»;  - ФИО: «Вашук Александра Юрьевна»;  - номер телефона: «+375299888007»;  - паспорт: «MP25211»;  - адрес: «Микромарс Д3»;  - почта: «Shaka@mail.ru».  2. Нажать на кнопку «Сохранить» | Ожидаемый результат: заказчик создан |
| Фактический результат: заказчик успешно создан. Результат представлен в приложении В на рисунке В.16. |

Четвертый тест будет проведён на форме «Посылка» для логистов.

Тест-кейс для тестирования функции добавления посылки будет представлен в таблицы 3.9

Таблица 3.9 – Тест-кейс функции добавления посылки

|  |  |  |
| --- | --- | --- |
| Функция | Шаги выполнения | Результат |
| Добавление посылки | 1.Заполнить поля данных:  - название: «Ящик икры черной "Самса"»;  - заказчик: «Вашук Александра Юрьевна»;  – дата с: «12.06.2024»;  - дата по: «27.06.2024»;  - машина: «Volvo FH 500»;  - водитель: «Гильмульдин Александрович Папов»;  - город отправки: «Минск»;  - город получения: «Пинск»;  - цена посылки: «2500».  2. Нажать на кнопку «Сохранить» | Ожидаемый результат: посылка отобразиться в таблице на форме «Главная» |
| Фактический результат: посылка успешно создана. Результат представлен в приложении В на рисунке В.17. |

Пятый тест будет проведён на форме «Документы».

Тест-кейс для тестирования функции создания договора будет представлен в таблицы 3.10

Таблица 3.10– Тест-кейс функции добавления заказчика

|  |  |  |
| --- | --- | --- |
| Функция | Шаги выполнения | Результат |
| Создания договора | 1.Заполнить поля данных:  - тип документа: «Word»;  - пользователь: «Абдусаломов Нурридин Угли»;  - посылка: «Холодильник LG GC-B509SBSM».  2. Нажать на кнопку «Распечатать» | Ожидаемый результат: создания документ заказа |
| Фактический результат: успешно создан документ заказа. Результат представлен в приложении В на рисунке В.18. |

Шестой тест будет проведён на форме «Документы».

Тест-кейс для тестирования функции создания отчета о всех посылках клиента будет представлен в таблицы 3.11

Таблица 3.11 – Тест-кейс функции создания отчета

|  |  |  |
| --- | --- | --- |
| Функция | Шаги выполнения | Результат |
| Создания отчёта | 1.Заполнить поля данных:  - тип документа: «Excel»;  - пользователь: «Абдусаломов Нурридин Угли»;  - дата с и по: «».  2. Нажать на кнопку «Распечатать» | Ожидаемый результат: создания отчета о заказах. |
| Фактический результат: успешно создан отчет о заказах Результат представлен в приложении В на рисунке В.19. |

Седьмой тест будет проведён на форме «Отметки».

Тест-кейс для тестирования функции отметки прибытия посылки будет представлен в таблицы 3.12

Таблица 3.12 – Тест-кейс функции отметки

|  |  |  |
| --- | --- | --- |
| Функция | Шаги выполнения | Результат |
| Создания отметки | 1.Заполнить поля данных:  - посылка: «Ящик икры черной "Самса"»  - тип отметки: «Ожидает» | Ожидаемый результат: обновление цены товара. |

Продолжение таблицы 3.12

|  |  |  |
| --- | --- | --- |
| Функция | Шаги выполнения | Результат |
|  | - дата задержки: «28.06.2024».  2. Нажать на кнопку «Сохранить» | Фактический результат: успешное обновление цены. Результат представлен в приложении В на рисунке В.20. |

Восьмой тест будет проведён на форме «Водители».

Тест-кейс для тестирования функции добавления водителя будет представлен в таблицы 3.13

Таблица 3.13 – Тест-кейс функции добавления водителя

|  |  |  |
| --- | --- | --- |
| Функция | Шаги выполнения | Результат |
| Добавления водителя | 1.Заполнить поля данных:  - ФИО: «Диденко Максим Максимович»  - номер телефона: «+375311113654»  - номер прав: «WR234RW»;  - дата рождения: «14.02.1990»  2. Нажать на кнопку «Сохранить» | Ожидаемый результат: создание водителя. |
| Фактический результат: успешное добавление водителя. Результат представлен в приложении В на рисунке В.21. |

Девятый тест будет проведён на форме «Транспорт».

Тест-кейс для тестирования функции создания автомобиля будет представлен в таблицы 3.14

Таблица 3.14 – Тест-кейс функции создания отчета

|  |  |  |
| --- | --- | --- |
| Функция | Шаги выполнения | Результат |
| Создания автомобиля | 1.Заполнить поля данных:  - марка автомобиля: «Mercedes-Benz G78»;  - водитель: «Диденко Максим Максимович»;  - максимальный вес: «8900»;  - максимальный объём: «7000»;  - номер машины «AA00B24».  2. Нажать на кнопку «Распечатать» | Ожидаемый результат: добавления автомобиля в базу. |
| Фактический результат: успешное создание автомобиля. Результат представлен в приложении В на рисунке В.22. |

# **Описание справочной системы**

Справочная система была разработана с использованием программного обеспечения Dr.Explain.

Справочная система включает в себя несколько ключевых разделов, которые были определены в предоставленном изображении:

* вход;
* регистрация;
* рабочее место логиста;
* работа с заказами;
* создание посылок;
* создание отметок;
* создание документации;
* рабочее место сотрудника транспортного отдела;
* работа с водителями;
* работа с транспортом.

Справочная система может быть вызвана из главной формы при нажатии на кнопку «Справка». Справка представлена на рисунке 3.2



Рисунок 3.3 – Справка

# **Применение**

# **Назначение программного средства**

Приложение «Sub-Zero.exe» предназначено для автоматизации транспортно-логистического предприятия. Программное средство представляет пользователю возможности для поиска, добавления, отображения, и редактирования имеющихся в базе объектов: посылок, клиентов, водителей, транспорта, уменьшает затраты времени используемого при составлении договора перевозки посылки, ведет учет посылок клиентов. Под эту роль был создан специальный макет, который позволит логисту быстро и удобно заключать договоры.

Основные разработанные функции:

* авторизация и регистрация работников в системе;
* возможность добавлять и редактировать заказчиков;
* возможность добавлять и редактировать посылки;
* возможность указывать дату получения/задержки посылки;
* возможность добавлять и редактировать водителей;
* возможность добавлять и редактировать транспортные средства;
* возможность экспорта данных в документы Microsoft Office.

Приложение «Sub-Zero.exe» разработано специально для логистических операций и не предназначено для использования в других отраслях, таких как производство или розничная торговля. Оно оптимизировано для работы с данными, связанными с перевозками, и может не подходить для управления данными, не относящимися к логистике. Кроме того, функциональность экспорта ограничена поддержкой документов Microsoft Office, что может не соответствовать потребностям компаний, использующих другие форматы для документооборота.

# **Условия применения**

Для применения данного программного средства необходимы следующие технические требования:

* процессор Intel Core i5 или выше;
* минимальный объём оперативной памяти 500 Мбайт;
* оперативная система Windows 11;
* платформа .Net Framework v4.8;
* видеокарта с объёмом памяти не менее 256 Мбайт;
* клавиатура, мышь, принтер.

Процесс установки программного средства «Sub-Zero.exe» включает следующие шаги:

* загрузка программы: скачайте установочный файл «Sub-Zero.exe» с github или носителя, предоставленного разработчиком;
* установка: запустите скачанный установочный файл и следуйте инструкциям мастера установки.

# **Заключение**

Проект “Sub-Zero.exe” представляет собой программное средство, разработанное для автоматизации рабочего места менеджера по работе с клиентами и сотрудника транспортного отдела в транспортно-логистической компании. Организационная сущность задачи заключается в создании инструмента, который облегчит работу логистов, менеджеров и водителей в области транспортной логистики. Предметная область проекта охватывает отслеживание статуса доставки, управление запасами, выбор транспортного средства, создание грузов и генерацию отчетов.

Технические требования для программы включают процессор Intel Core i5 или выше, минимальный объем оперативной памяти 500 Мбайт, операционную систему Windows 11, платформу .Net Framework v4.8 и видеокарту с объемом памяти не менее 256 Мбайт. Логическая организация данных включает описание структуры данных и связей между объектами (посылки, клиенты, водители, транспорт). Физическая организация данных включает выбор СУБД и оптимизацию запросов. Справочная система описывает функциональность и предоставляет помощь пользователям.

Программное средство Sub-Zero предоставляет возможности поиска, добавления, отображения и редактирования объектов в базе данных. Входные данные включают информацию о заказчиках, посылках, водителях и транспорте. Выходные данные могут быть отчетами, статусами доставки и другой информацией.

Среди преимуществ разработанного ПС можно выделить автоматическое управление запасами, гибкость и масштабируемость, а также интуитивный пользовательский Макет. Однако следует учитывать ограниченность функциональности экспорта данных в документы Microsoft Office и то, что программа не подходит для управления данными, не связанными с транспортной-логистикой.

Sub-Zero представляет собой важный инструмент для транспортно-логистических компаний, помогая им оставаться конкурентоспособными в современном бизнес-окружении.

# **Список использованных источников**

1. Багласова, Т.Г. Методические указания по оформлению курсовых проектов, дипломных проектов и отчетов для учащихся специальности 2-40 01 01 «Программное обеспечение технологий» / С.В. Банцевич, Т.Г. Багласова. – Минск: КБП, 2022. – 49 c.
2. Михалевич, В.Ю. Методические указания к курсовому проектированию для обучающихся специальности 2-40 01 01 «Программное обеспечение информационных технологий – Минск: КБП, 2023. – 3 c.
3. ГОСТ Р 54694-2011. Логистика. Термины и определения. – Введ. 2013-01-01. – М.: Стандартинформ, 2012.;
4. Текст программы. Требования к содержанию, оформлению и контролю качества: ГОСТ 19.401-2000. – Введ. 01.09.2001. – Минск : Межгос. совет по стандартизации, метрологии и сертификации, 2000. – 16 с.
5. 9 лучших программ для автоматизации рабочих процессов [2024 г.] [Электронный ресурс] – Режим доступа: https://nauchniestati.ru/spravka;
6. Автоматизация рабочего места – RPA ROBIN [Электронный ресурс] – Режим доступа: https://www.atlassian.com/ru/agile;
7. Автоматизированное рабочее место (АРМ): что это? [Электронный ресурс] – Режим доступа: https://www.liga-commersantov.ru/articles;
8. Обязанности менеджера по работе с клиентами – чем занимается и как [Электронный ресурс] – Режим доступа: https://sales-generator.ru/blog;
9. Разработка автоматизированного рабочего места менеджера [Электронный ресурс] – Режим доступа: https://spravochnick.ru/informatika;
10. Топ лучших приложений для управления персоналом для автоматизации [Электронный ресурс] – Режим доступа: https://www.guru99.com/ru/best;
11. ТОП-26 программ мониторинга и контроля работы сотрудников за [Электронный ресурс] – Режим доступа: https://habr.com/ru/articles;
12. Удобство и эффективность: разработка автоматизированного рабочего места [Электронный ресурс] – Режим доступа: <https://appmaster.io/ru/blog>;
13. Что значит АРМ (Автоматизированное рабочее место)? [Электронный ресурс] – Режим доступа: https://alexrovich.ru/info;
14. [Visual Studio 2022 IDE - Programming Tool for Software Developers (microsoft.com)](https://visualstudio.microsoft.com/vs/) [Электронный ресурс] – Режим доступа: https://visualstudio.microsoft.com/vs/.
15. [Visual Studio 2022 IDE - Programming Tool for Software Developers (microsoft.com)](https://visualstudio.microsoft.com/vs/) [Электронный ресурс] – Режим доступа: https://visualstudio.microsoft.com/vs/.
16. C Sharp – Википедия [2024 г.] [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/C\_Sharp;
17. [Microsoft Office – Wikipedia](https://en.wikipedia.org/wiki/Microsoft_Office) [2024 г.] [Электронный ресурс] – Режим доступа: https://en.wikipedia.org/wiki/Microsoft\_Office;
18. [Microsoft Visio – Википедия (wikipedia.org)](https://ru.wikipedia.org/wiki/Microsoft_Visio) 2024 г.] [Электронный ресурс] – Режим доступа: <https://ru.wikipedia.org/wiki/Microsoft_Visio>;
19. Windows 11 [Электронный ресурс]. – Microsoft, 2024. – Режим доступа: https://support.microsoft.com/ru-ru/meetwindows11. – Дата доступа: 16.03.2024;

# **ПРИЛОЖЕНИЕ А**

**(обязательное)**

**Текст программных модулей**

static public bool AddDriver(Drivers driver)

{

try

{

if (driver == null)

{

throw new ArgumentException("Объект Driver не может быть null.");

}

if (string.IsNullOrEmpty(driver.FullName) || string.IsNullOrEmpty(driver.PhoneNumber) || string.IsNullOrEmpty(driver.DrivingLicenseNumber) || driver.DateOfBirth == null)

{

throw new ArgumentException("Поля FullName, PhoneNumber, DrivingLicenseNumber и DateOfBirth не могут быть пустыми или равными null.");

}

if (driver.PhoneNumber.Length != 13)

{

throw new ArgumentException("Номер телефона должен содержать цифр.");

}

if (driver.DrivingLicenseNumber.Length < 6 || !driver.DrivingLicenseNumber.All(char.IsLetterOrDigit))

{

throw new ArgumentException("Водительское удостоверение должно содержать не менее 6 цифр или букв.");

}

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "INSERT INTO driever (fio, phoneNumber, driverLessons, dateOfBirth) VALUES (@fio, @phoneNumber, @driverLessons, @dateOfBirth)";

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@fio", driver.FullName);

command.Parameters.AddWithValue("@phoneNumber", driver.PhoneNumber);

command.Parameters.AddWithValue("@driverLessons", driver.DrivingLicenseNumber);

command.Parameters.AddWithValue("@dateOfBirth", driver.DateOfBirth);

command.ExecuteNonQuery();

}

}

return true;

}

catch (ArgumentException ex)

{

MessageBox.Show(ex.Message);

return false;

throw;

}

} static public bool DeleteCar(Car car)

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "DELETE FROM Car WHERE id = @id";

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@id", car.id);

command.ExecuteNonQuery();

}

}

return true;

} public static void LoadCustomersToDataGrid(System.Windows.Controls.DataGrid dgCust)

{

List<Customer> customers = new List<Customer>();

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "SELECT \* FROM Customer";

using (SqlCommand command = new SqlCommand(query, connection))

{

using (SqlDataReader reader = command.ExecuteReader())

{

while (reader.Read())

{

int id = Convert.ToInt32(reader["id"]);

string fio = reader["fio"].ToString();

string numberOfPhone = reader["numberOfPhone"].ToString();

string pochta = reader["pochta"].ToString();

int isUrFace = Convert.ToInt32(reader["isUrFace"]);

string pasport = reader["pasport"].ToString();

string adress = reader["adress"].ToString();

string ynp = reader["ynp"].ToString();

string nameCompany = reader["nameCompany"].ToString();

string adressCompany = reader["adressCompany"].ToString();

int rastSchet;

int.TryParse(reader["rastSchet"].ToString(), out rastSchet);

string nameBank = reader["nameBank"].ToString();

string codeBank = reader["codeBank"].ToString();

Customer customer = new Customer(id, fio, numberOfPhone, pochta, isUrFace, pasport, adress, ynp, nameCompany, adressCompany, rastSchet, nameBank, codeBank);

customers.Add(customer);

}

}

}

}

dgCust.ItemsSource = customers;

} static public bool AddCustomerUr(Customer cust)

{

try

{

if (cust == null)

{

throw new ArgumentException("Объект Customer не может быть null.");

}

if (string.IsNullOrEmpty(cust.Fio) || string.IsNullOrEmpty(cust.NumberOfPhone) || string.IsNullOrEmpty(cust.Pochta) || string.IsNullOrEmpty(cust.Adress))

{

throw new ArgumentException("Поля Fio, NumberOfPhone, Pochta и Adress не могут быть пустыми или равными null.");

}

if (cust.NumberOfPhone.Length != 13)

{

throw new ArgumentException("Номер телефона должен содержать 12 цифр.");

}

if (!cust.Pochta.EndsWith("@mail.ru") && !cust.Pochta.EndsWith("@gumail.ru"))

{

throw new ArgumentException("Почта должна быть @mail.ru или @gumail.ru.");

}

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string insertQuery = "INSERT INTO Customer ( fio, numberOfPhone, pochta, isUrFace, pasport, adress, ynp, nameCompany, adressCompany, rastSchet,nameBank ,codeBank ) VALUES (@Fio,@NumberOfPhone,@Pochta,@IsUrFace,@Pasport,@Adress,@Ynp,@NameCompany,@AdressCompany,@RastSchet ,@NameBank ,@CodeBank)";

using (SqlCommand insertCommand = new SqlCommand(insertQuery, connection))

{

insertCommand.Parameters.AddWithValue("@Fio", cust.Fio);

insertCommand.Parameters.AddWithValue("@NumberOfPhone", cust.NumberOfPhone);

insertCommand.Parameters.AddWithValue("@Pochta", cust.Pochta);

insertCommand.Parameters.AddWithValue("@IsUrFace", cust.IsUrFace);

insertCommand.Parameters.AddWithValue("@Pasport", cust.Pasport ?? (object)DBNull.Value);

insertCommand.Parameters.AddWithValue("@Adress", cust.Adress);

insertCommand.Parameters.AddWithValue("@Ynp", cust.Ynp ?? (object)DBNull.Value);

insertCommand.Parameters.AddWithValue("@NameCompany", cust.NameCompany ?? (object)DBNull.Value);

insertCommand.Parameters.AddWithValue("@AdressCompany", cust.AdressCompany ?? (object)DBNull.Value);

insertCommand.Parameters.AddWithValue("@RastSchet", cust.RastSchet);

insertCommand.Parameters.AddWithValue("@NameBank", cust.NameBank ?? (object)DBNull.Value);

insertCommand.Parameters.AddWithValue("@CodeBank", cust.CodeBank ?? (object)DBNull.Value);

int rowsAffected = insertCommand.ExecuteNonQuery();

return rowsAffected > 0;

}

}

}

catch (ArgumentException ex)

{

MessageBox.Show(ex.Message);

return false;

throw;

}

}

public static void GenerateWordReport(Customer selectedCustomer, Box box, Drivers driver)

{

using (WordprocessingDocument wordDocument = WordprocessingDocument.Create("C:\\Users\\kupit\\OneDrive\\Desktop\\Отчет.docx", WordprocessingDocumentType.Document))

{

MainDocumentPart mainPart = wordDocument.AddMainDocumentPart();

mainPart.Document = new Document();

Body body = mainPart.Document.AppendChild(new Body());

AppendParagraph(body, "ДОГОВОР ПЕРЕВОЗКИ ГРУЗА", true, JustificationValues.Center);

AppendParagraph(body, $"{DateTime.Now:dd.MM.yyyy}.", false, JustificationValues.Right);

AppendParagraph(body, "Наименование компании: Sub\_Zero", false, JustificationValues.Right);

AppendParagraph(body, "1. СТОРОНЫ ДОГОВОРА", true);

AppendParagraph(body, $"1.1. Перевозчик: {driver.FullName}");

AppendParagraph(body, $"1.2. Грузополучатель: {selectedCustomer.Fio}");

AppendParagraph(body, $"1.3. Грузоотправитель: {selectedCustomer.Fio}");

AppendParagraph(body, "2. ХАРАКТЕРИСТИКИ ГРУЗА", true);

AppendParagraph(body, $"2.1. Вес: {box.Weight}");

AppendParagraph(body, $"2.2. Объем: {box.Value}");

AppendParagraph(body, "3. ПУНКТ ОТПРАВЛЕНИЯ И НАЗНАЧЕНИЯ", true);

AppendParagraph(body, $"3.1. Пункт отправления: {GetCityIdByName(box.SityOtprav)}");

AppendParagraph(body, $"3.2. Пункт назначения: {GetCityIdByName(box.SityDostav)}");

AppendParagraph(body, "4. СТОИМОСТЬ ДОСТАВКИ И ПОРЯДОК РАСЧЕТОВ", true);

AppendParagraph(body, $"4.1. Стоимость доставки: {box.PriceBox – box.Price}");

AppendParagraph(body, "5. СРОКИ ПЕРЕВОЗКИ", true);

AppendParagraph(body, $"5.1. Дата отправки: {box.dateOtprav}");

AppendParagraph(body, $"5.2. Дата получения: {box.datePolych}");

AppendParagraph(body, "6. ПРАВА И ОБЯЗАННОСТИ СТОРОН", true);

AppendParagraph(body, "6.1. Перевозчик обязуется доставить груз в указанное место в указанные сроки. Грузоотправитель обязуется оплатить услуги перевозки в соответствии с договором. Перевозчик не несет ответственности за потерю или повреждение груза.");

AppendParagraph(body, "7. ОТВЕТСТВЕННОСТЬ ЗА НАРУШЕНИЕ УСЛОВИЙ СДЕЛКИ", true);

AppendParagraph(body, "7.1. В случае нарушения условий договора стороны несут ответственность в соответствии с действующим законодательством.");

AppendParagraph(body, "8. ФОРС-МАЖОР", true);

AppendParagraph(body, "8.1. В случае наступления обстоятельств непреодолимой силы, стороны освобождаются от ответственности за неисполнение обязательств по договору.");

if (selectedCustomer.IsUrFace == 1)

{

AppendParagraph(body, "9. ЮРИДИЧЕСКИЕ АДРЕСА И БАНКОВСКИЕ РЕКВИЗИТЫ СТОРОН", true);

AppendParagraph(body, $"9.1. Юридический адрес: {selectedCustomer.AdressCompany}");

AppendParagraph(body, $"9.2. Банковские реквизиты: Банк – {selectedCustomer.NameBank}, Код банка – {selectedCustomer.CodeBank}");

}

AppendParagraph(body, "10. ПОДПИСИ СТОРОН", true);

AppendParagraph(body, "Грузоотправитель (Подпись)\_\_\_\_\_\_\_\_\_\_\_\_\_");

AppendParagraph(body, "Директор (Подпись)\_\_\_\_\_\_\_\_\_\_\_\_\_");

mainPart.Document.Save();

}

}

static public bool AddUser(string username, string password, string position)

{

try

{

if (string.IsNullOrEmpty(username) || string.IsNullOrEmpty(password) || string.IsNullOrEmpty(position))

{

throw new ArgumentException("Имя, пароль, или тип работника не могут быть пустыми");

}

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string checkQuery = "SELECT COUNT(\*) FROM users WHERE nickname = @Username";

using (SqlCommand checkCommand = new SqlCommand(checkQuery, connection))

{

checkCommand.Parameters.AddWithValue("@Username", username);

int userCount = (int)checkCommand.ExecuteScalar();

if (userCount > 0)

{

return false;

}

}

string insertQuery = "INSERT INTO users (nickname, password, kindOfWork) VALUES (@Username, @Password, @Position)";

using (SqlCommand insertCommand = new SqlCommand(insertQuery, connection))

{

insertCommand.Parameters.AddWithValue("@Username", username);

insertCommand.Parameters.AddWithValue("@Password", password);

insertCommand.Parameters.AddWithValue("@Position", position);

insertCommand.ExecuteNonQuery();

}

}

return true;

}

catch (ArgumentException ex)

{

MessageBox.Show(ex.Message);

return false;

throw;

}

}

public static Drivers GetDriver(int id)

{

Drivers driver = null;

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "SELECT \* FROM driever WHERE id = @id";

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@id", id);

using (SqlDataReader reader = command.ExecuteReader())

{

if (reader.Read())

{

driver = new Drivers(

id: reader.GetInt32(reader.GetOrdinal("id")),

fullName: reader.GetString(reader.GetOrdinal("fio")),

phoneNumber: reader.GetString(reader.GetOrdinal("phoneNumber")),

drivingLicenseNumber: reader.GetString(reader.GetOrdinal("driverLessons")),

dateOfBirth: reader.GetDateTime(reader.GetOrdinal("dateOfBirth"))

);

}

}

}

}

return driver;

}

static public bool DeleteCustomer(Customer customer)

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "DELETE FROM Customer WHERE id = @id";

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@id", customer.Id);

command.ExecuteNonQuery();

}

}

return true;

}

# **ПРИЛОЖЕНИЕ Б**

**(справочное)**

**Формы выходных документов**

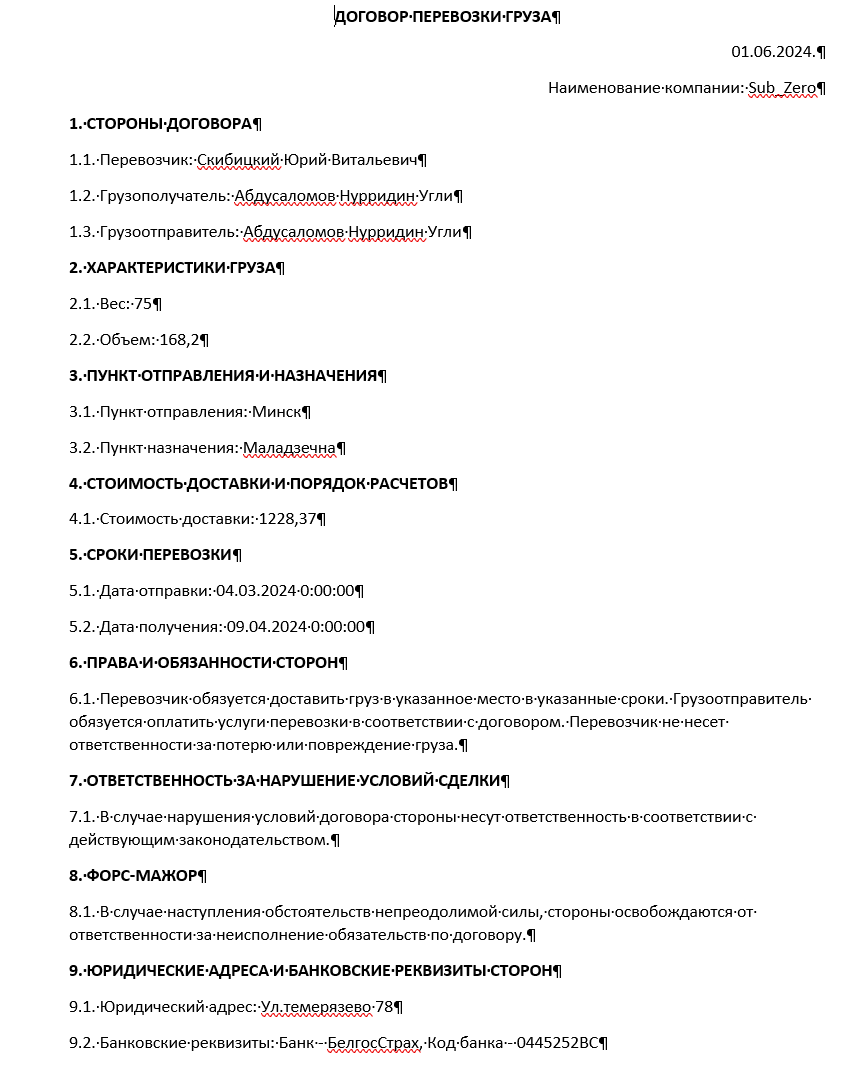


Рисунок Б.1 – Договор перевозки

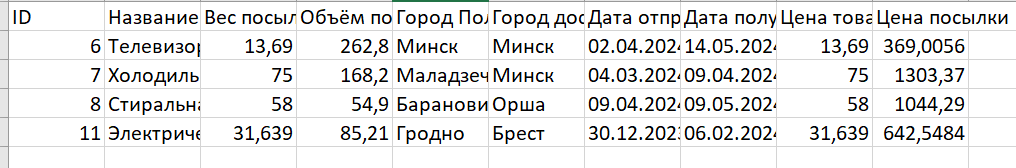


Рисунок Б.2 – Отчет о посылках пользователя

# **П**ы**РИЛОЖЕНИЕ В**

**(справочное)**

**Результаты работы программы**

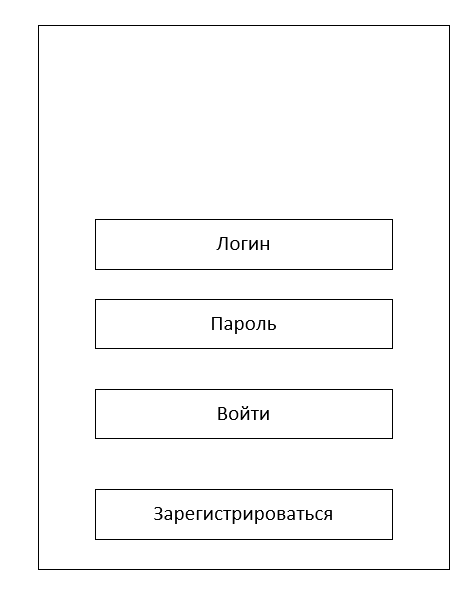


Рисунок В.1 – Макет входа

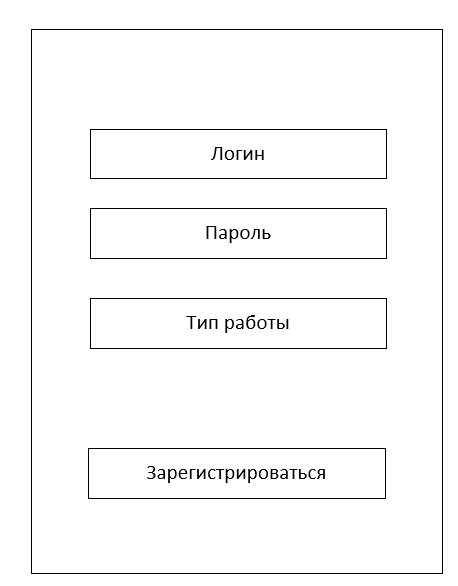


Рисунок В.2 – Макет регистрации

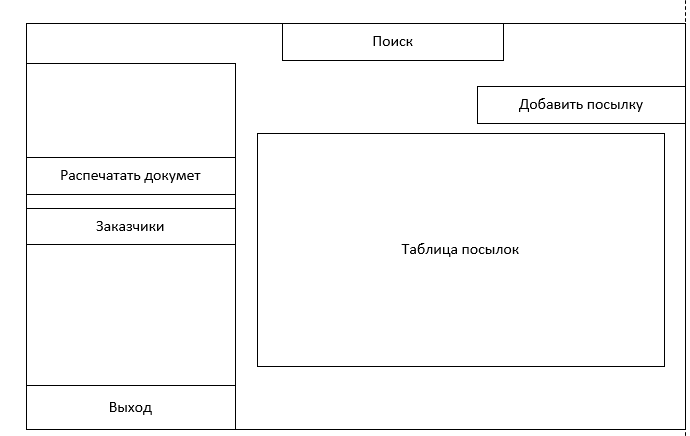


Рисунок В.3 – Макет «Главная» для логистов

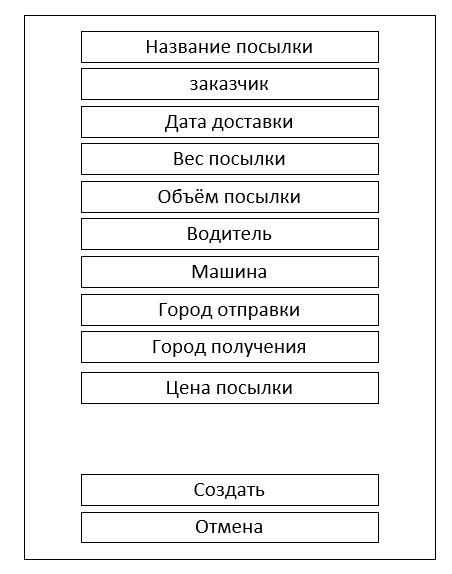


Рисунок В.4 – Макет добавления посылки

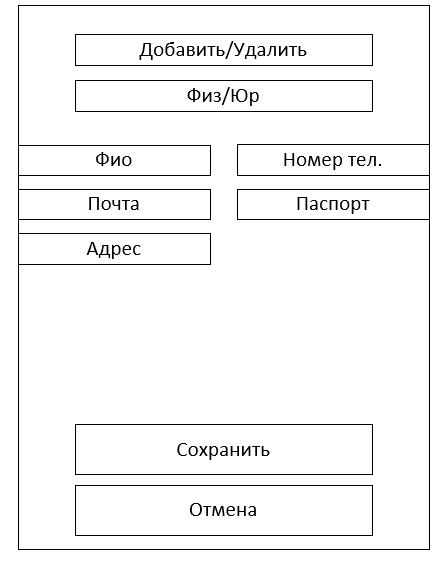


Рисунок В.5– Макет заказчиков для физ.лиц



Рисунок В.6– Макет заказчиков для юр.лиц

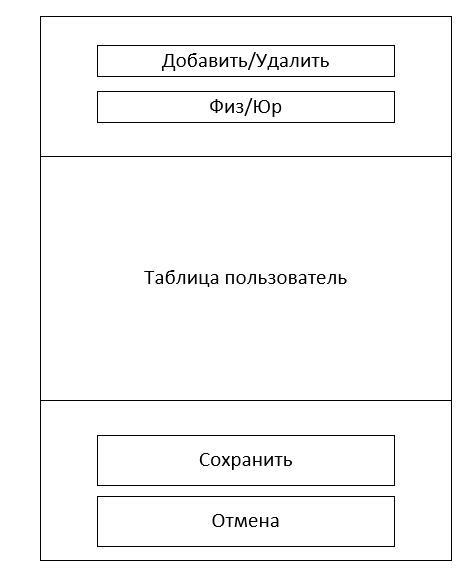


Рисунок В.7– Макет заказчиков

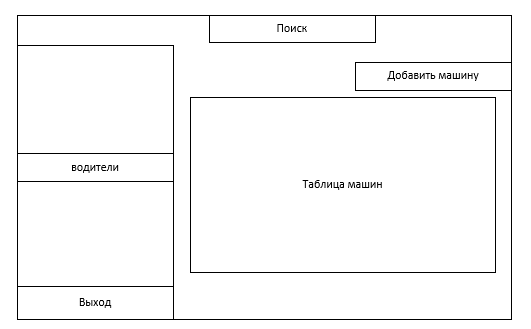


Рисунок В.8 – Макет «Главная» для сотрудников тс организации

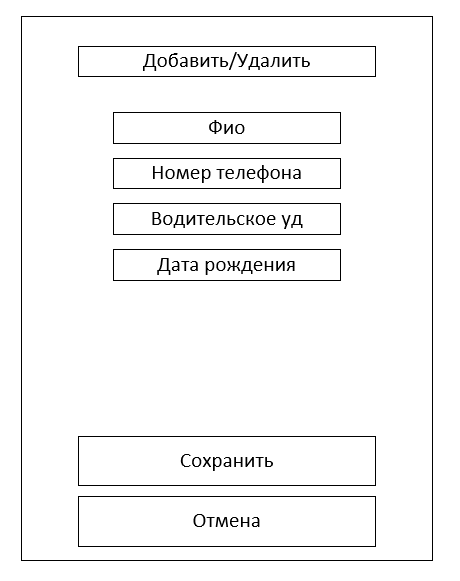


Рисунок В.9 – Макет водителей

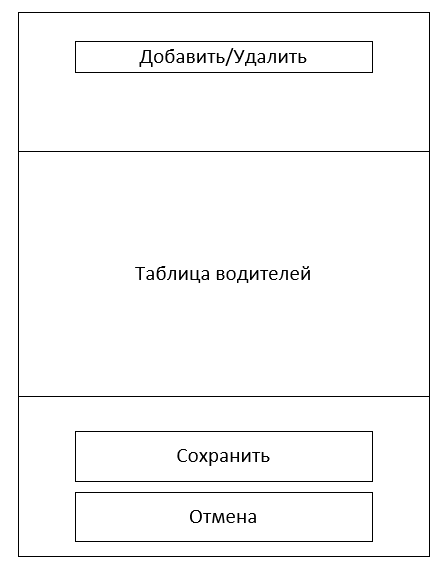


Рисунок В.10 – Макет водителей

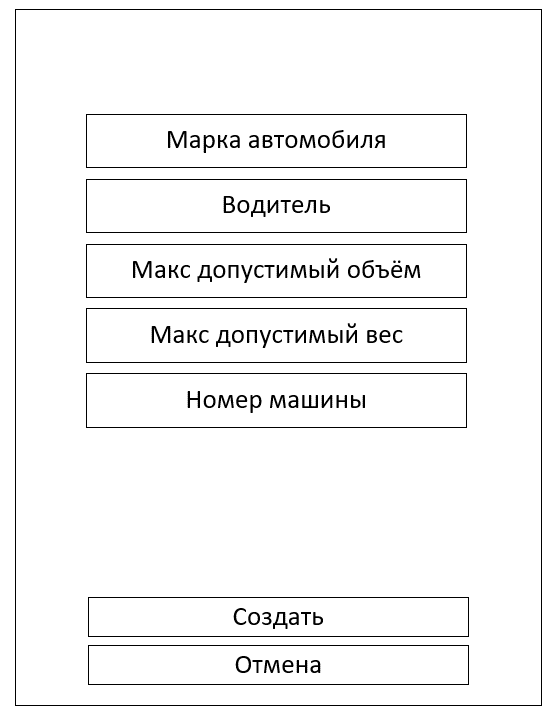


Рисунок В.11– Макет машин

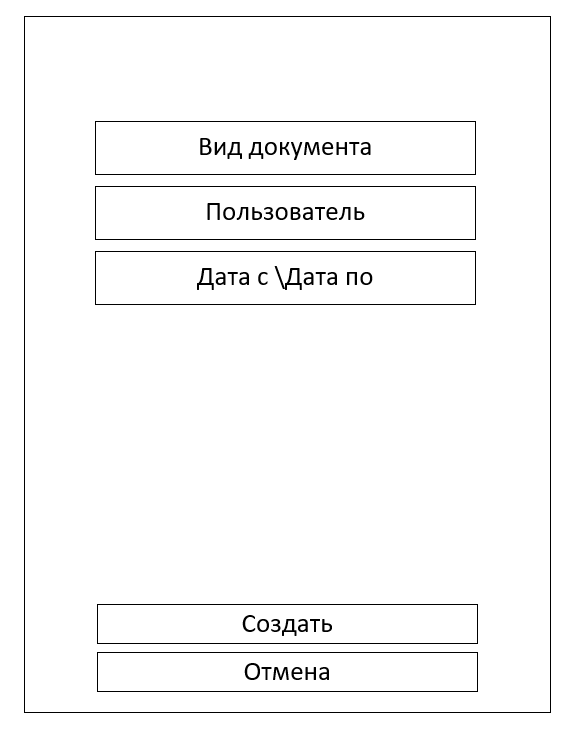


Рисунок В.12 – Макет документации

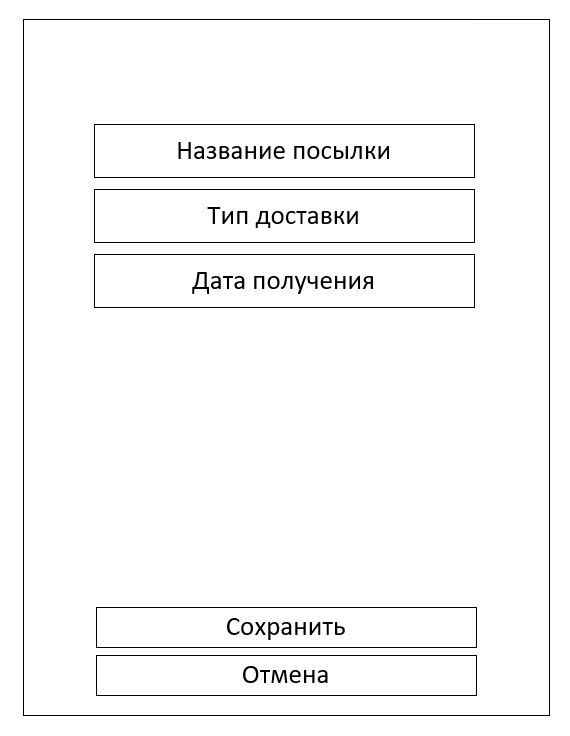


Рисунок В.13 – Макет меток посылки

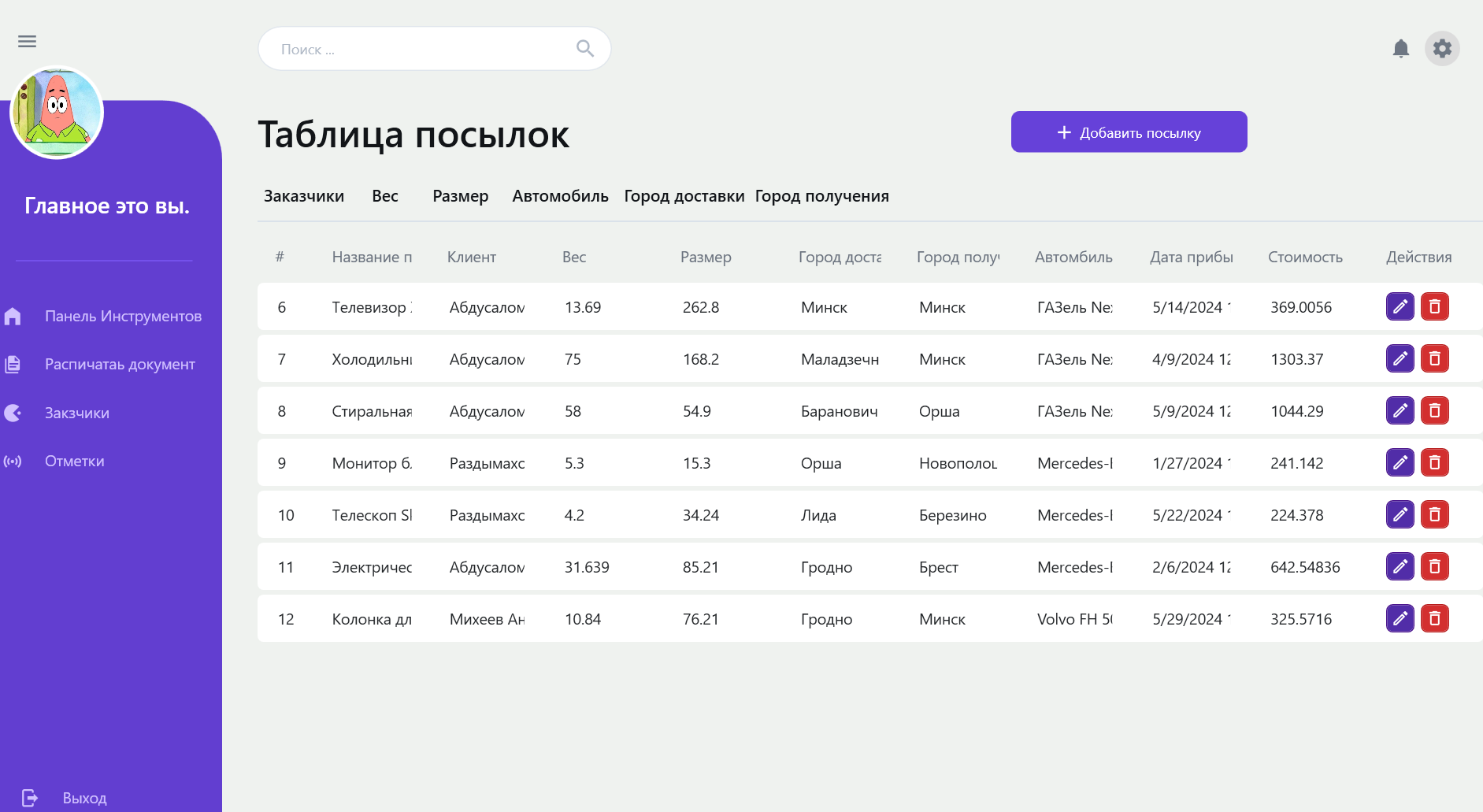


Рисунок В.14 – Результат входа логиста

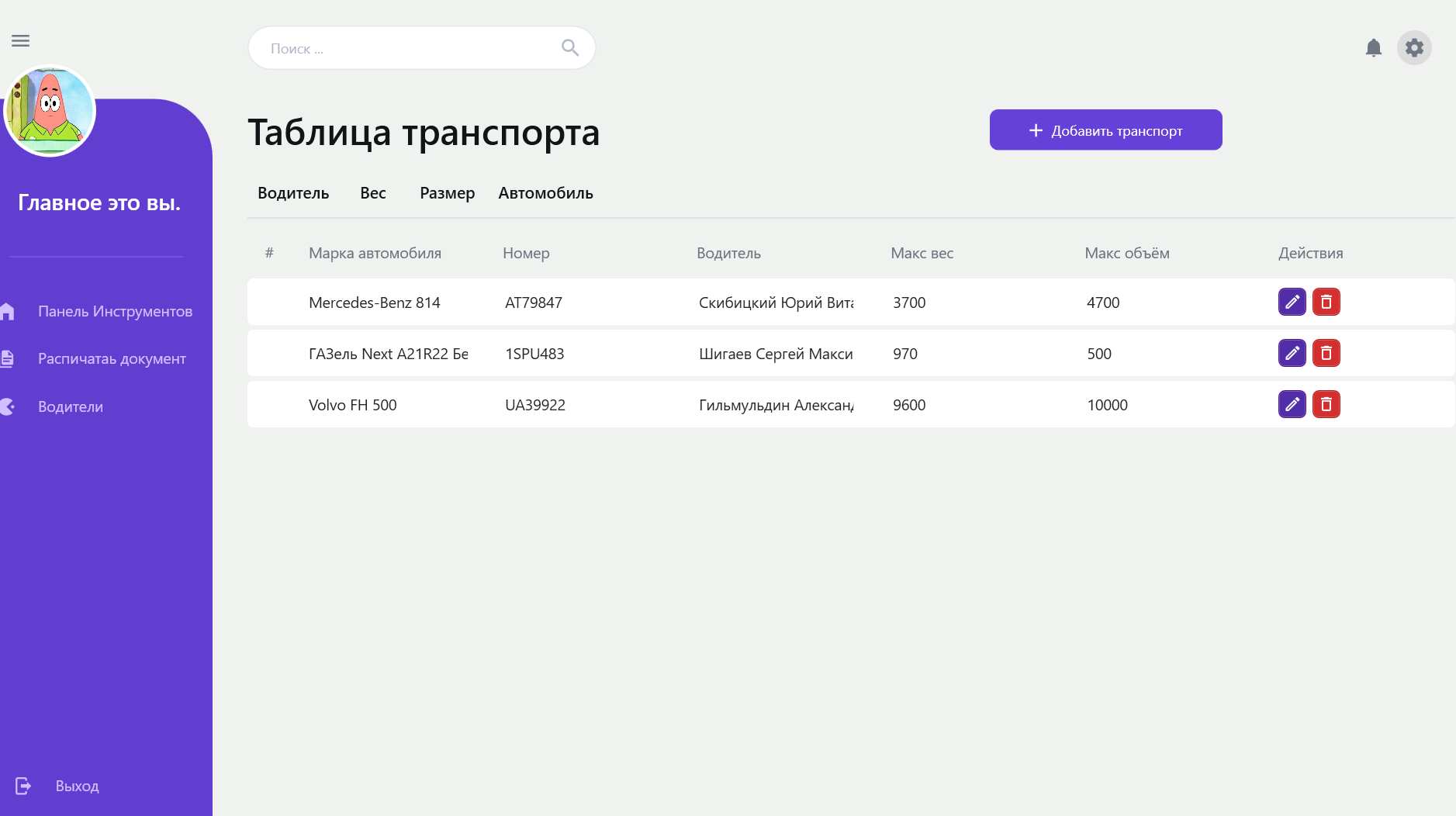


Рисунок В.15 – Результат входа сотрудника транспортного отдела

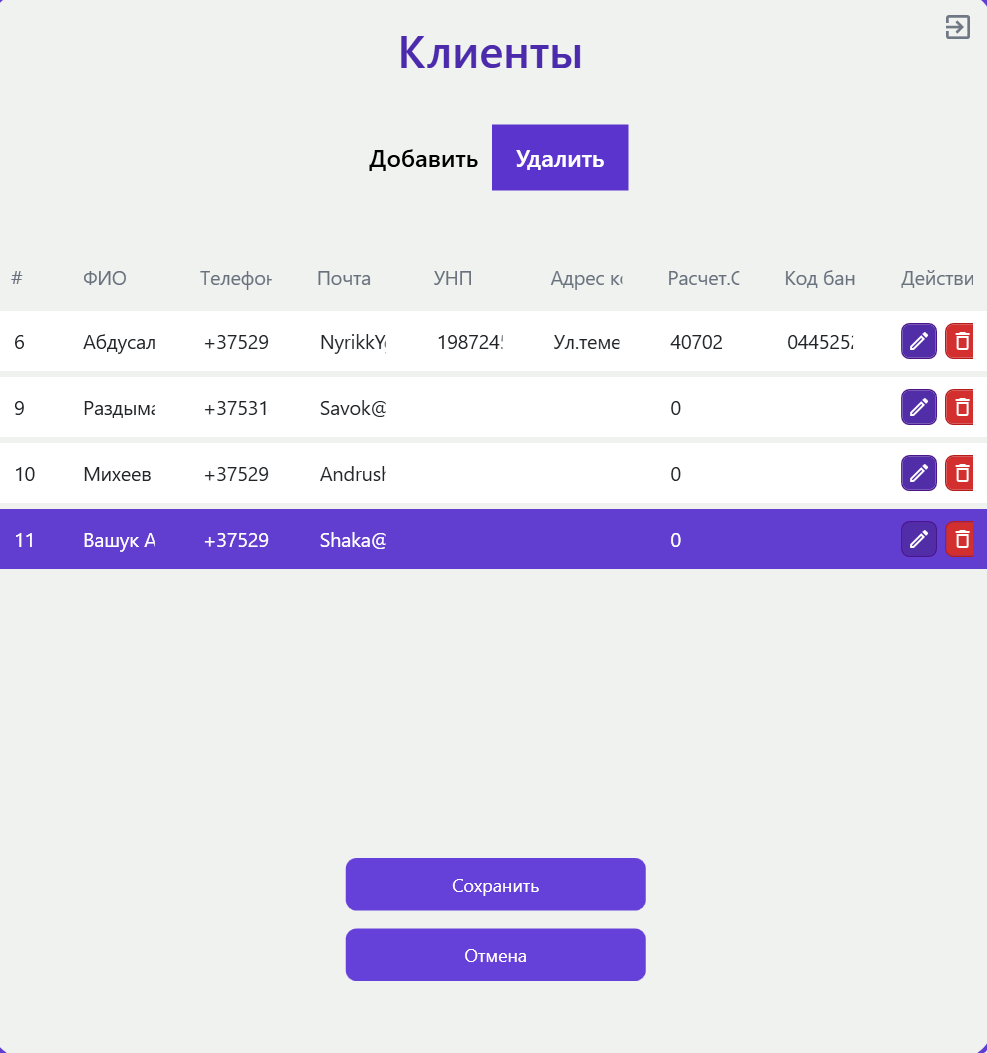


Рисунок В.16 – Результат добавления клиента

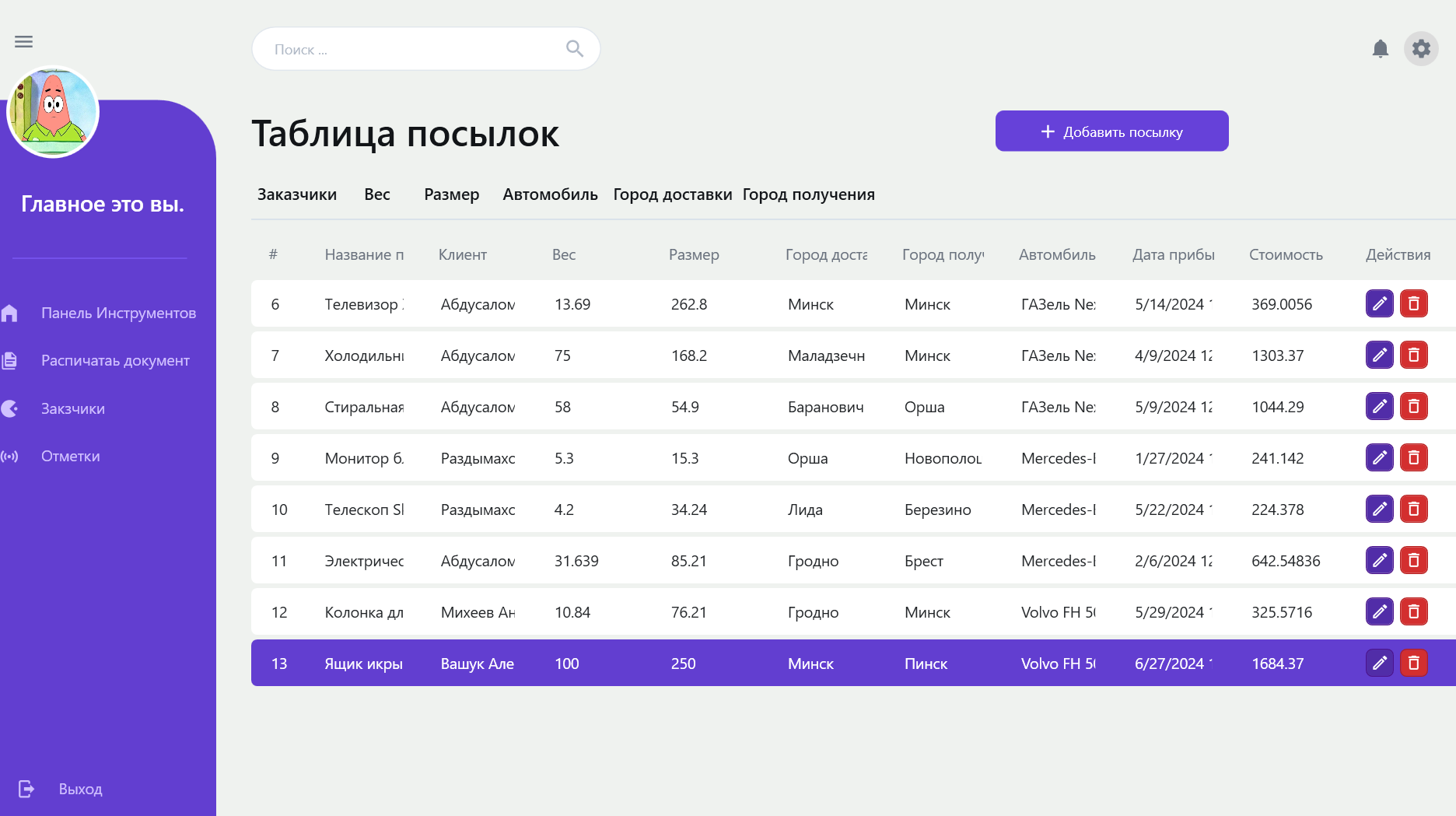


Рисунок В.17 – Результат создания посылки

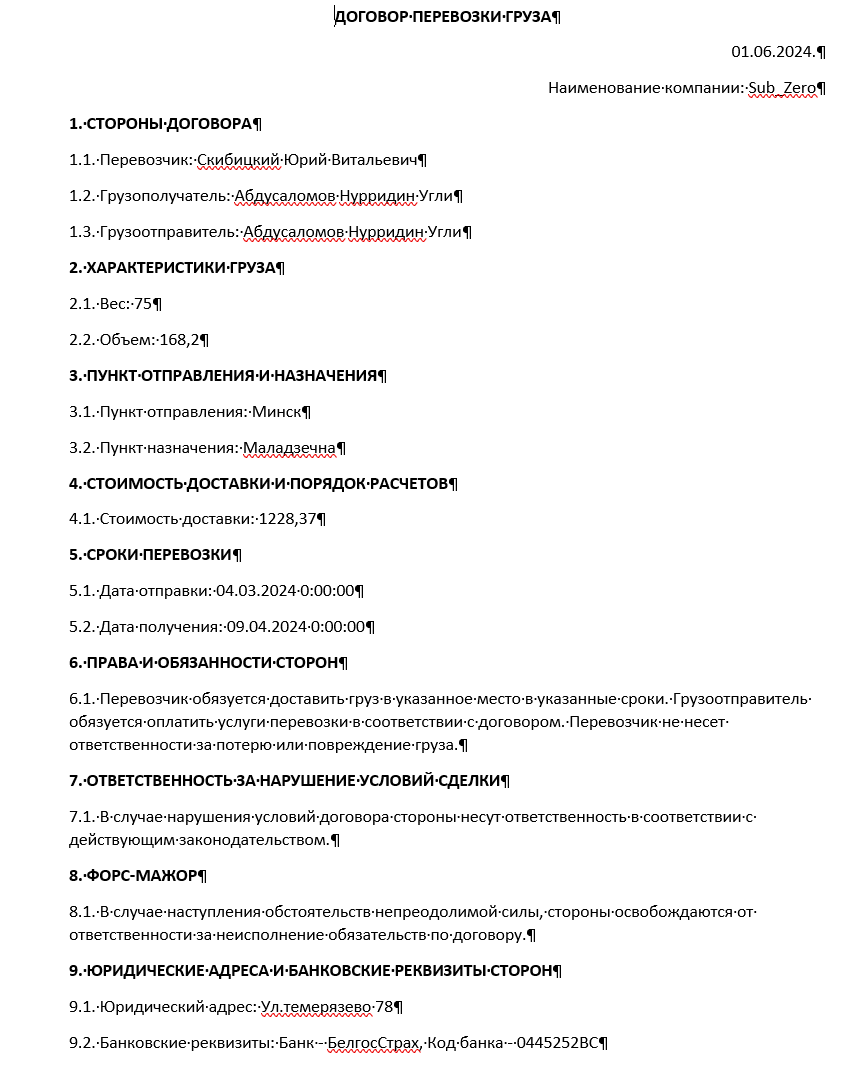


Рисунок В.18 – Результат создания договора

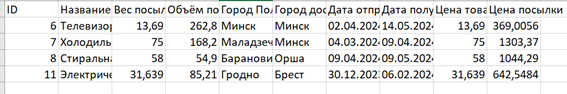


Рисунок В.19 – Результат создания отчета по всем посылках

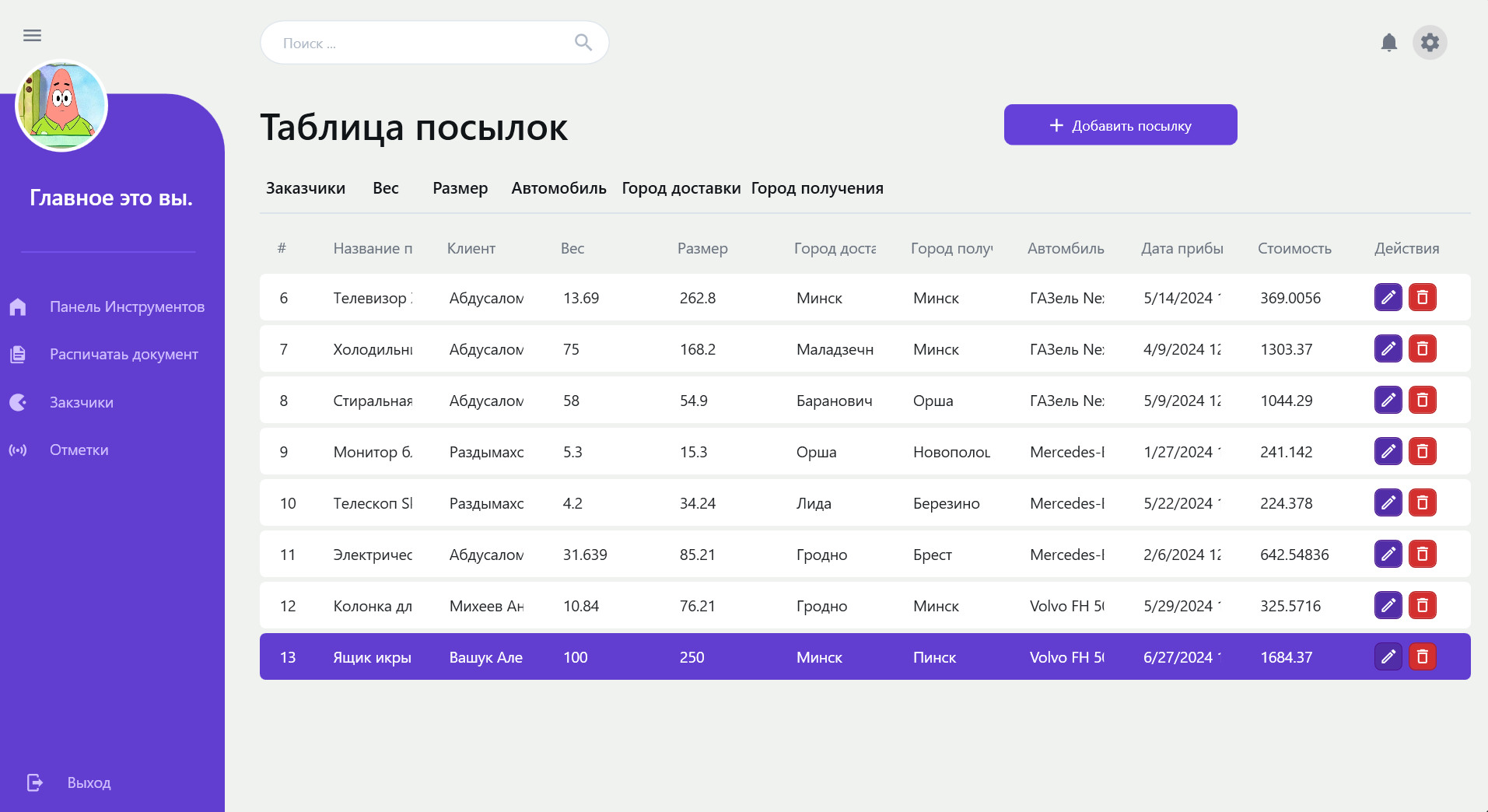


Рисунок В.20 – Результат отметки

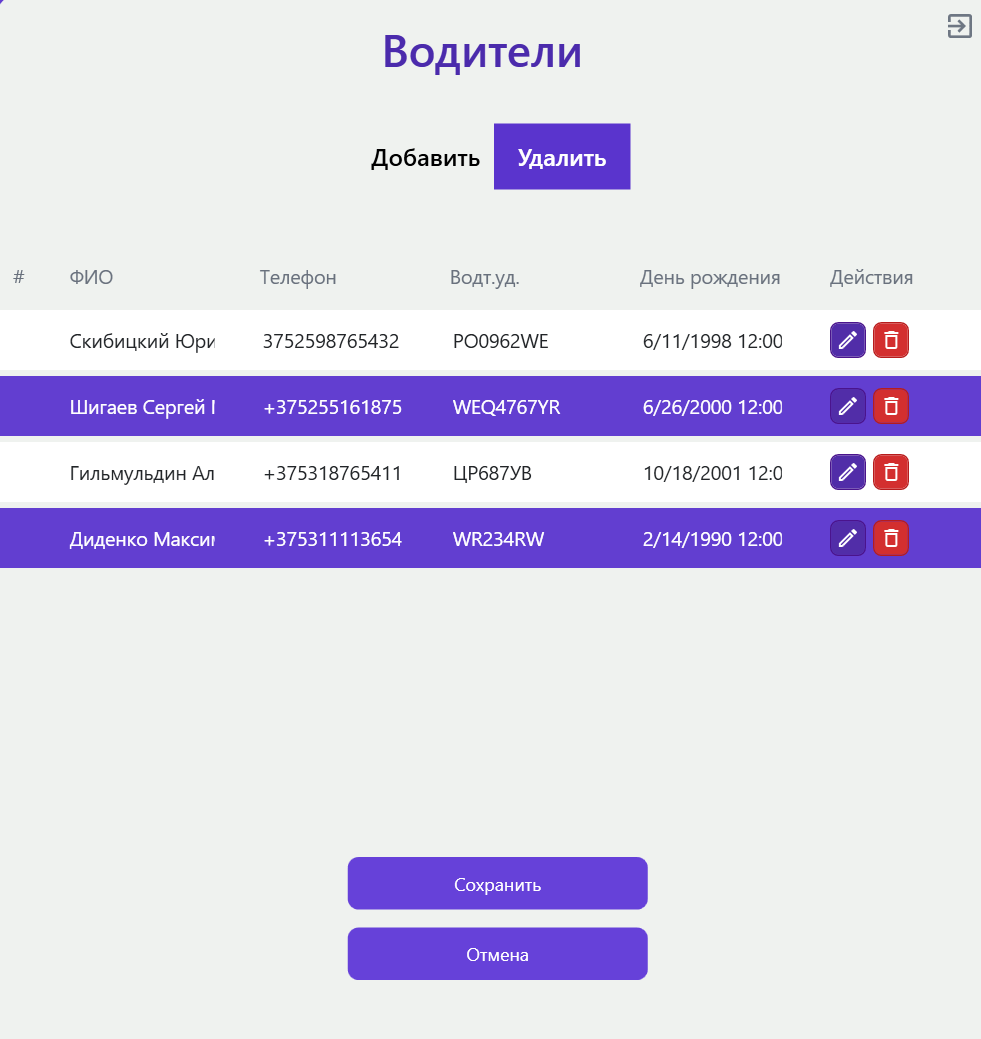


Рисунок В.21 – Результат добавления водителя

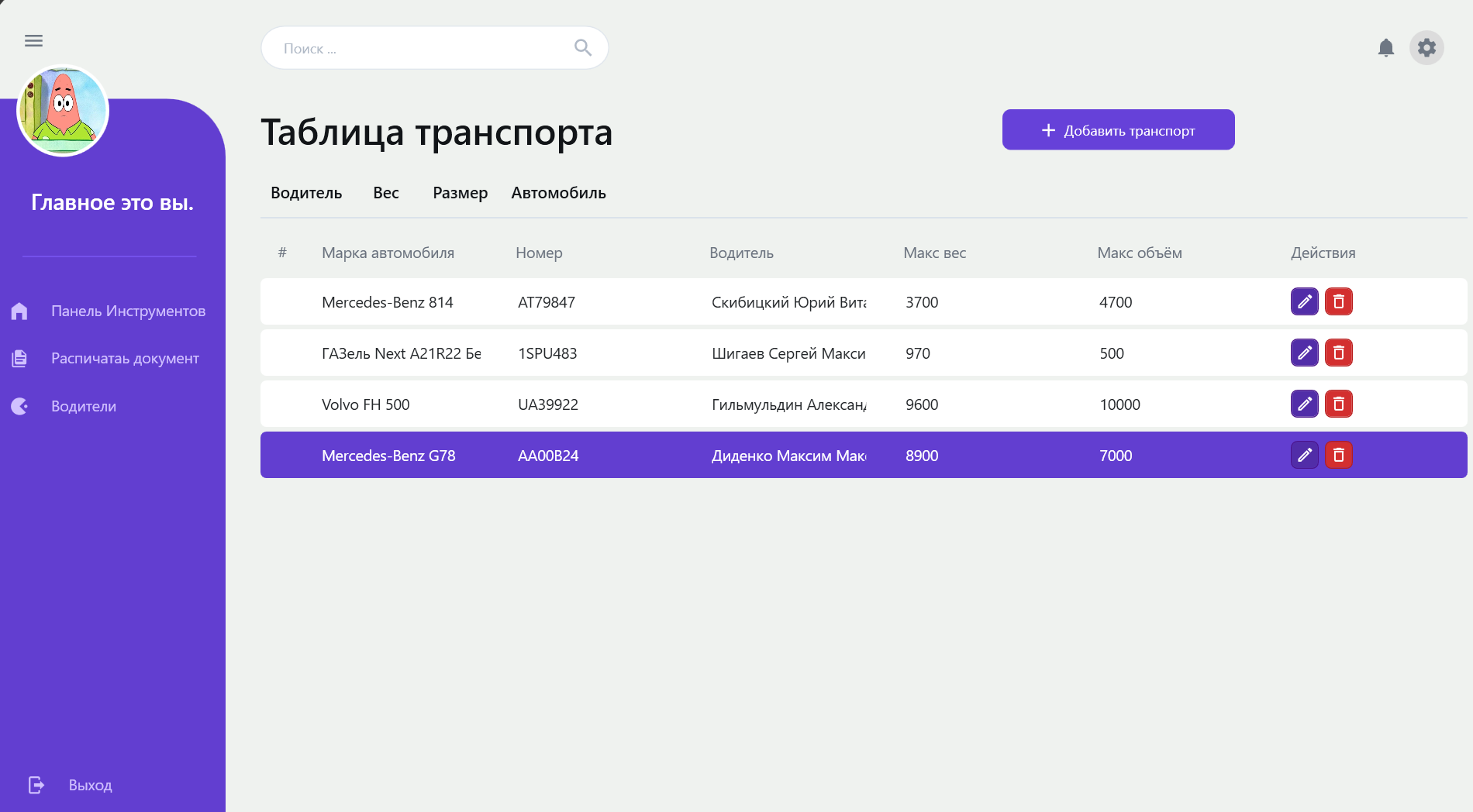


Рисунок В.22 – Результат добавления транспорта