**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**Федеральное государственное автономное образовательное учреждение высшего образования**

**«Санкт-Петербургский государственный политехнический университет Петра Великого»**

**(ФГАОУ ВО «СПбПУ»)**

**Институт среднего профессионального образования**

**КУРСОВОЙ ПРОЕКТ**

09.02.07 Информационные системы и программирование

специальность

отделение Информационных Технологий

По дисциплине: МДК 02.01 Технология разработки программного обеспечения

Тема Разработка АИС для городского автовокзала

Исполнитель: студент(ка) IV курса группы 42919/4

Подосенов Тихон Романович

(фамилия, имя, отчество)

Руководитель преподаватель ИСПО

(ученная степень, звание)

Челищева Лилия Дмитриевна

(фамилия, имя, отчество)

|  |  |
| --- | --- |
| Курсовой проект допущен к защите  «12» апреля 2025 г.  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Защитил(а) курсовой проект с оценкой  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  «12» апреля 2025 г. |

2025

**Министерство науки и высшего образования Российской Федерации**

федеральное государственное автономное образовательное учреждение

высшего образования

«Санкт-Петербургский политехнический университет Петра Великого»

(ФГАОУ ВО «СПбПУ»)

Институт среднего специального образования

УТВЕРЖДЕНО   
предметной комиссией  
 «Информационные системы и программирование»  
   
Председатель ПЦК   
\_\_\_\_\_\_\_\_\_\_\_\_ Сынкова А.Д.

**ЗАДАНИЕ НА КУРСОВОЕ ПРОЕКТИРОВАНИЕ**

по модулю ПМ.02 «Осуществление интеграции программных модулей»

Специальность 09.02.07 «Информационные системы и программирование»

Студенту группы 42919/4 Подосенову Тихону Романовичу

Тема курсового проекта: Разработка АИС для городского автовокзала

**Индивидуальное задание**

*\_Проанализировать предметную область, сформировать спецификацию требований, спроектировать систему (разработать UML диаграммы Вариантов использования, последовательности, активности, состояний), разработать проект интерфейса пользователя, разработать Веб-приложение, используя технологии MVC, протестировать полученную программу, опубликовать полученное решение на хостинге.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

Дата выдачи задания 13.01.2025 Зав. отделением \_\_\_\_\_\_\_\_\_Назаров Д.А.

Срок окончания 12.04.2025 Преподаватель \_\_\_\_\_\_\_\_\_\_ Челищева Л.Д.

Студент \_\_\_\_\_\_\_\_\_\_\_ Подосенов Т.Р.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc195359791)

[1 Общая часть 4](#_Toc195359792)

[1.1 Описание предметной области 4](#_Toc195359793)

[1.2 Анализ требований 4](#_Toc195359794)

[1.3 Обзор средств программирования 9](#_Toc195359795)

[1.4 Вывод по общей части 10](#_Toc195359796)

[2 Специальная часть 11](#_Toc195359797)

[2.1 Постановка задачи 11](#_Toc195359798)

[2.2 Архитектура и описание структуры разрабатываемого приложения 13](#_Toc195359799)

[2.3 Проектирование пользовательского интерфейса 16](#_Toc195359800)

[2.4 Проектирование БД 22](#_Toc195359801)

[2.5 Разработка программы 25](#_Toc195359802)

[2.6 Тестирование программы 26](#_Toc195359803)

[ЗАКЛЮЧЕНИЕ 29](#_Toc195359804)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 30](#_Toc195359805)

[Приложение А (справочное) Результат работы программы 31](#_Toc195359806)

[Приложение Б (справочное) Исходный код программы 44](#_Toc195359807)

[Приложение В (справочное) Код Unit-тестов 66](#_Toc195359808)

[Приложение Г (справочное) Тест-кейсы 68](#_Toc195359809)

# **ВВЕДЕНИЕ**

Ежедневно на городском автовокзале отправляются и прибывают автобусы. Трафик людей в таких местах довольно высок и для обработки и систематизации информации по данным о рейсах, билетах, транспорте, сотрудниках, перевозчиках требуется информационная система, которая облегчит работу данного места.

Для решения данной проблемы был разработан курсовой проект на тему «Городской автовокзал». Целью данной работы стало создание информационной системы городского автовокзала для автоматизации учёта бронирования, продаж и возврата билетов на поездки по междугородным автобусным маршрутам, а также структурирования информации о перевозчиках, водителях, автобусах, маршрутах и расписании движения транспорта. Задачи проекта включают:

1. Проектирование структуры информационной системы: построение структуры базы данных, макета информационной системы и архитектуры проекта;
2. Выбор стека технологий;
3. Разработка модулей информационной системы;
4. Тестирование модулей информационной системы;
5. Сборка исходников проекта в одну директорию;
6. Написание отчета.
7. **Общая часть**

### Описание предметной области

Городской автовокзал является местом, где отправляются и прибывают автобусы. Трафик людей в таких местах довольно высок и для обработки и систематизации информации по данным о рейсах, билетах, транспорте, сотрудниках, перевозчиках требуется информационная система, которая облегчит работу данного места.

### Анализ требований

Для анализа предметной области был проведен опрос заинтересованных лиц:

* кассир;
* системный администратор.

Таблица 1 – Интервью с кассиром

| **Вопрос** | **Ответ** |
| --- | --- |
| Имя | Смирнова Елена Александровна |
| Место работы | Городской автовокзал |
| Должность | Кассир |
| Основные задачи | Бронирование и оформление билетов, работа с пассажирами. |
| Что должна уметь делать программа? | Давать возможность бронировать и оформлять билет, просматривать имеющихся перевозчиков, водителей, расписание, маршруты, приобретенные и оформленные билеты. |
| Какие данные должны вноситься при оформлении билета пассажиру? | ФИО пассажира, его возраст, серия и номер паспорта, номер рейса по расписанию, место в автобусе, наличие животных и метка о выкупе билета. |

Продолжение таблицы 1

|  |  |
| --- | --- |
| **Вопрос** | **Ответ** |
| Какие правила по бронированию и покупке билета? | При оформлении билета пассажир должен сказать, когда он хочет приобрести этот билет: сейчас или же потом, тогда этот билет будет забронирован на его имя. |
| Что если человек не успел на свой рейс, либо он потерял свой билет? | Стоимость билета в таком случае не возвращается, замещающий документ в виде справки не выдается, посадка пассажира на автобус не производится. |

Таблица 2 – Интервью с системным администратором

|  |  |
| --- | --- |
| **Вопрос** | **Ответ** |
| Имя | Иванов Дмитрий Сергеевич |
| Место работы | Городской автовокзал |
| Должность | Системный администратор |
| Основные задачи | Управление серверами, обеспечение безопасности и стабильности работы информационных систем, поддержка пользователей. |
| Где вы хотите хранить данные о билетах, пассажирах, рейсах и т.д.? | На нашем сервере. |
| С какой системой управления базами данных вы работаете или предпочли бы работать? | Я могу работать со многими базами, например: Oracle Database, PostgreSQL, SQLite и MySQL. Однако, со всеми ними я могу работать лишь на начальном уровне, поэтому мне бы подошла MS SQL Server, которую я хорошо знаю. |
| Данные о скольких людях ваш сервер должен хранить и обрабатывать в моменте максимально? | Думаю, минимум 600 человек. Максимум 700. |

Продолжение Таблицы 2

|  |  |
| --- | --- |
| **Вопрос** | **Ответ** |
| Как будет добавляться информация на программный продукт? | Наше руководство заключает договор с перевозчиками на план рейсов. При получении данного плана я добавляю информацию в базу данных, откуда она уже будет взята для остальных программных продуктов. |
| Какая информация от перевозчиков добавляется в базу данных? | Номер рейса, откуда и куда, дата и время отбытия и прибытия, количество мест и их расположение в автобусе, автомобильный номер автобуса. |

Актуальность ПО:

Создание программного обеспечения (ПО) является актуальным вопросом для городского автовокзала из-за увеличения числа пассажиров и усложнения процессов бронирования и оформления билетов через бумажный документооборот. Введение ПО позволит существенно оптимизировать работу автовокзала, повысить эффективность и точность обработки данных.

Выявленные потребности:

* Учет и анализ данных пассажиров. Сбор и анализ информации о пассажирах, включая их предпочтения и историю поездок.
* Получение полной информации о заказах. Возможность быстро и легко получать всю информацию о бронированных и приобретенных билетах.
* Управление клиентской базой (добавление/изменение данных). Легкое управление данными пассажиров, включая добавление и изменение информации.
* Оформление и корректировка заказов. Удобное оформление и корректировка билетов, в том числе при необходимости внесения изменений.
* Надежная защита персональных и финансовых данных. Обеспечение высокого уровня безопасности для хранения и обработки персональных и финансовых данных пассажиров.

Ключевые аспекты:

* Автоматизация процессов продаж: Полная автоматизация процессов бронирования и оформления билетов, что сокращает время на обслуживание пассажиров.
* Эффективная работа с пассажирами: Улучшение качества обслуживания пассажиров за счет быстрого доступа к информации и удобного интерфейса.
* Безопасное хранение информации: Надежное хранение всех данных, включая персональные.

Для более подробного анализа требований была составлена диаграмма идентификации точек зрения, представленная на рисунке 1.



Рисунок – Диаграмма идентификации точек зрении

В таблице 3 указаны требования, соотнесенные с точками зрения.

Таблица 3 – Требования, соотнесенные с точками зрения

| **Кассир** | **Системный администратор** |
| --- | --- |
| Покупка и бронирование билетов | Просмотр забронированных и купленных билетов |
| Просмотр забронированных и купленных билетов | Просмотр расписания |
| Просмотр расписания | Просмотр перевозчиков |
| Просмотр перевозчиков | Добавление, изменение и удаление рейсов |
| Просмотр водителей | Добавление, удаление и изменение всех данных в БД, кроме билетов |
| Просмотр маршрутов | Просмотр маршрутов |
| Просмотр пунктов назначения | Просмотр водителей |
|  | Просмотр пунктов назначения |

Так же были выведены пользовательские истории (“use case”), которые приведены далее:

1. Я как кассир, хочу иметь доступ к билетам, чтобы осуществлять оформление и бронирование билетов для пассажира;
2. Я как кассир, хочу знать какие есть перевозчики, чтобы информировать людей о том, с кем они поедут;
3. Я как кассир, хочу иметь возможность принять у пассажира билет обратно, чтобы вернуть ему деньги за него;
4. Я как системный администратор, хочу просматривать информацию по приобретенным и забронированным билетам, чтобы предоставлять ее пассажирам и кассирам в программном продукте;
5. Я как системный администратор, хочу просматривать информацию по рейсам, чтобы работать с ней (добавлять/изменять/удалять);
6. Я как системный администратор, хочу просматривать информацию по перевозчикам, чтобы составлять данные по рейсам;

### Обзор средств программирования

Для разработки информационной системы городского автовокзала рассматривались следующие технологии:

* + 1. **C# (.NET Framework)**

Преимущества:

* полная интеграция с Windows-платформой;
* богатая стандартная библиотека;
* возможность создания как десктопных (WinForms, WPF), так и веб-приложений.

Недостатки:

* ориентированность преимущественно на экосистему Microsoft [1];
  + 1. **Python (PyQt/Tkinter)**

Преимущества:

* простота разработки;
* богатые возможности для работы с данными.

Недостатки:

* более низкая производительность;
* сложности с распространением готовых приложений [2].
  + 1. **Итоговый стек**

Для реализации проекта была выбрана следующий стек разработки:

* язык программирования: C#;
* среда разработки: Visual Studio 2022 [3];
* GUI-фреймворк: Windows forms;
* СУБД: База данных: SQL Server Management Studio 20 [4].

### Вывод по общей части

Разрабатываемая информационная система позволит оптимизировать работу компании, минимизировать ошибки, связанные с ручным учётом, и повысить общую эффективность бизнеса. Она будет способствовать повышению качества обслуживания клиентов, улучшению аналитики и принятия управленческих решений, а также обеспечит надёжное хранение и быстрый доступ к данным.

1. **Специальная часть**

### Постановка задачи

Постановка задачи выполнена в соответствии со стандартами Единой Системы Программной документации и руководящими отраслевыми методическими материалами ГОСТ 19.201-78.

* + 1. **Основания для разработки**

Основанием для разработки является задание в рамках курсового проекта Челищевой Л.Д., представителя ФГАОУ ВО «СПбПУ» Института среднего профессионального образования, а также требования к автоматизации работы туристического агентства, выявленные в ходе анализа предметной области.

* + 1. **Назначение разработки**
       1. **Функциональное назначение**

Данная система предназначена для автоматизации процессов управления городским автовокзалом. Она позволяет:

* Выбирать рейсы с детализацией информации (номер рейса, дата отправления, время в пути, количество мест).
* Осуществлять покупку и бронирование билетов.
* Редактировать данные из базы данных.
* Обеспечивать безопасное хранение данных.
* Поддерживать масштабируемость системы и высокую доступность.

Система должна быть удобной для использования сотрудниками таких ролей, как кассирам и системным администраторам.

* + - 1. **Нефункциональное назначение**

Нефункциональные требования определяют условия, в которых должна работать система, и обеспечивают её надёжность, удобство использования, безопасность и масштабируемость. Ниже приведены основные нефункциональные требования к разрабатываемой информационной системе:

Производительность:

* Система должна обеспечивать низкую задержку при обработке запросов, особенно при высокой нагрузке.
* Оптимизация базы данных для быстрого выполнения операций чтения и записи.

Аутентификация и авторизация:

* Разграничение прав доступа в зависимости от роли пользователя (кассир, системный администратор).

Удобство использования:

* Время отклика системы на действия пользователя (например, нажатие кнопки или выполнение запроса) не должно превышать 10 секунд.
  + 1. **Требование к программе или программному изделию**
       1. **Требования к составу и параметрам технических средств**

Приложение может использоваться на устройствах, которые имеют:

* Объем свободной оперативной памяти ~ 8 Гб.
* Объем необходимой памяти на жестком диске ~ 10 Гб.
* Скорость подключения к интернету – не менее 100 Мбит/сек.

### Архитектура и описание структуры разрабатываемого приложения

* + 1. **Бизнес-логика работы информационной системы городского автовокзала**

Разработанная диаграмма вариантов использования представлена на рисунке 2.



Рисунок – Диаграмма вариантов использования

* + 1. **Бизнес-логика добавления билета в базу данных**

На рисунке 3 показана диаграмма последовательности добавления билета в базу данных.



Рисунок – Диаграмма последовательности

* + 1. **Состояние объекта в процессе работы информационной системы**

На рисунке 4 отображена диаграмма состояний объекта «Рейс».



Рисунок – Диаграмма состояний объекта «Тур»

* + 1. **Бизнес-логика процесса информационной системы**

На рисунке 5 приложена диаграмма процесса удаления маршрута системным администратором.



Рисунок – Диаграмма активности

### Проектирование пользовательского интерфейса

* + 1. **Карта навигации приложения**

Карта навигации приложения представлена на рисунке 7. На рисунке 7 представлена схема карты навигации для системного администратора.



Рисунок – Карта навигации приложения для системного администратора

На рисунке 8 представлена схема карты навигации для кассира.



Рисунок – Карта навигации приложения для кассира

* + 1. **Интерфейс вход в приложение**

Для наглядного представления пользовательского интерфейса ниже приведены WireFrames (скелетные макеты) [5]. Каждый элемент описан с указанием его функциональности и назначения (см. рисунок 9–14).

На рисунке 9 представлен интерфейс входа в приложение. В данном меню пользователь может совершить вход с помощью своего логина и пароля



Рисунок – WireFrame эскиз окна авторизации



Рисунок – Mockup эскиз формы авторизации

Основные элементы:

* поле для ввода логина (Логин);
* поле для ввода пароля (Пароль), которое скрывает вводимые символы;
* кнопка "Войти" для отправки данных в систему обработки данных.

Цель: обеспечить безопасный вход пользователя в систему с

проверкой его ролей и прав доступа.

* + 1. **Главный раздел**

На рисунке 11-12 представлен интерфейс главного меню, который отображается после успешной авторизации пользователя



Рисунок – WireFrame эскиз окна главного меню



Рисунок – Mockup эскиз формы главного раздела

Основные элементы:

* Перевозчики: открывает раздел с данными о перевозчиках и кнопкой для их изменения;
* Водители: открывает раздел с данными о водителях и кнопкой для их изменения;
* Маршруты: открывает раздел с данными о маршрутах и кнопкой для их изменения;
* Расписание: открывает раздел с данными о расписании и кнопкой для его изменения;
* Билеты: открывает раздел с данными о билетах и кнопкой для из изменения;
* Автобусы: открывает раздел с данными об автобусах и кнопкой для из изменения;
* Выход: переносит на раздел авторизации.

Цель: предоставить удобный способ перехода между разделами системы.

* + 1. **Просмотр информации**

На рисунке 12 – 13 представлен интерфейс для отображения данных о билетах



Рисунок – WireFrame эскиз окна формы «Билеты»



Рисунок – Mockup эскиз раздела «Билеты»

Этот экран позволяет просматривать информацию о билетах, а также предоставляет возможность выгрузить эти данные:

* Выгрузить в TXT: позволяет выгружать таблицу в текстовый файл;
* На главную: переводит пользователя на главный раздел.

Цель: обеспечить удобный просмотр данных из базы данных.

### Проектирование БД

На рисунке 14 была разработана ER-модель базы данных для информационной системы городского автовокзала.



Рисунок – ER-модель базы данных

* + 1. **Словарь данных**

В таблице 4 представлен словарь данных для базы данных, используемой в приложении.

Таблица 4 – Таблица «roles»

| **Сущность** | **Поля и их свойства** |
| --- | --- |
| Buses | IDBus (PK),  Carrier (FK),  BusNumber,  NumberOfSeats,  FireExtinguisher (NULL) |
| Carriers | IDCarrier (PK),  Name |
| Destinations | IDDestination (PK),  City,  Street,  House |
| Drivers | IDDriver (PK),  Surname,  Name,  Patronymic,  Age (NULL),  YearsOfDrivingExperience |
| Routes | IDRoute (PK),  PlaceOfDeparture (FK),  PlaceOfArrival (FK),  TimeOfDeparture,  TravelTimeInHours |
| Autorization | IDAut(PK),  Role(FK),  Login,  Password |
| Roles | IDRole(PK),  Role |

Продолжение таблицы 4

|  |  |
| --- | --- |
| **Сущность** | **Поля и их свойства** |
| Tickets | IDTicket(PK),  Surname,  Name,  Patronimyc,  Age,  PassportSeriesAndNumber,  Schedule(FK),  PlaceNumber,  Pets (NULL),  BoughtOut |

На основе словаря данных была созданных база данных в СУБД Microsoft SQL Server 2022 (Рисунок 15)

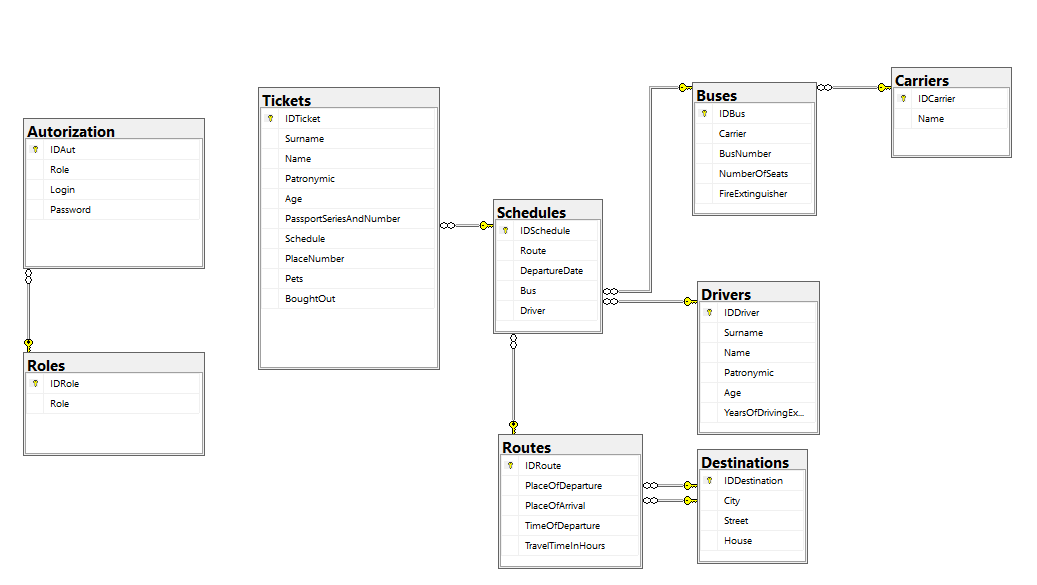


Рисунок – Реализованная база данных

### Разработка программы

* + 1. **Диаграмма классов**

Для функционирования базы данных с приложением была создана модель при помощи Entity Framework. Модель представлена на рисунке 16.

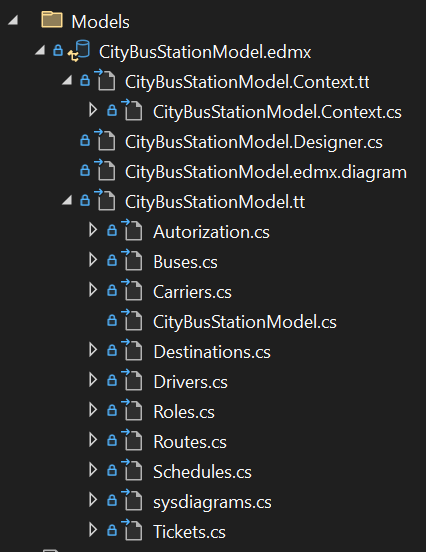


Рисунок – Модель базы данных

После был создан главный класс Form1, в котором был прописан весь основной функционал. Результат работы программы и диаграмма класса представлены в приложении А.

* + 1. **Текст программы**

Текст программы в соответствии с ГОСТ 19.101-77 (СТ СЭВ 1626-79) и ГОСТ 19.401-79 (СТ СЭВ 3746-82) представляет собой запись программы на исходном языке программирования с необходимыми комментариями [6]. Текст программы приведен в приложении Б.

* + 1. **Реализация пользовательского интерфейса приложения**

При разработке пользовательского интерфейса особое внимание уделялось удобству взаимодействия с системой для каждого типа пользователей (системный администратор и кассир) [7].

### Тестирование программы

Для тестирования была выбрана стратегия модульного тестирования, которая позволяет автоматически получать результат о текущем состоянии тестируемого кода. Модульные тесты обеспечивают проверку корректности работы ключевых функций каждой формы. Текущие модульные тесты приведены на рисунке. Код, реализующий данные тесты, будет предоставлен в Приложении В.

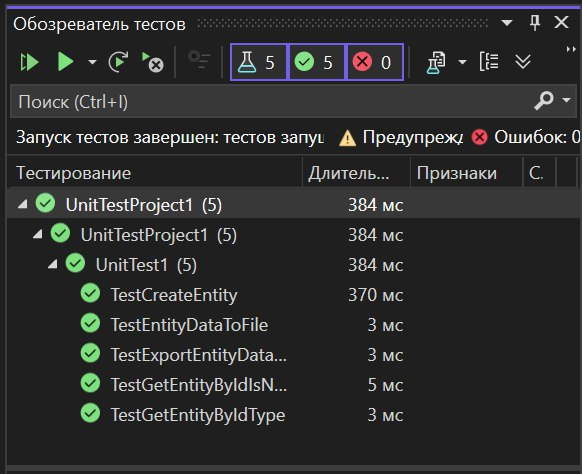


Рисунок – Модульные тесты

* + 1. **Модульные тесты для работы пользователя с формой**

Форма работы с клиентами (From1) отвечает за добавление, выгрузку данных, получение данных по ID. Для этой формы также было создано 5 тестов, которые проверяют следующие сценарии.

Первый модульный тест ("TestExportEntityDataToFile") проверяет, что данные успешно экспортируются в текстовый файл.

Второй модульный тест ("TestEntityDataToFile") проверяет, что для экспорта данных выбирается верный путь к текстовому файлу.

Третий модульный тест ("TestGetEntityByIdIsNotNull") проверяет, что результат получения записи по ID не пустой.

Четвертый модульный тест ("TestGetEntityByIdType") проверяет, что результат получения данных по ID соответствует определенному типу данных.

Пятый модульный тест ("TestCreateEntity") проверяет, что новые записи создаются правильно без потерь данных и ложных типов данных.

Все тесты были успешно выполнены, и их результаты подтверждают корректность работы следующих функций.

* + 1. **Написание тест-кейсов**

Для тестирования были разработаны тест-кейсы. После этого был проведен каждый тест кейс отдельно, чтобы зафиксировать результат. Тест-кейсы и результаты приведены в приложении Г.

# **ЗАКЛЮЧЕНИЕ**

Результатом курсового проекта является разработанное приложение для автоматизации работы городского автовокзала. Разработанная система направлена на оптимизацию ключевых бизнес-процессов компании, включая управление продажами билетов и клиентской базой. Она соответствует всем требованиям, сформулированным в техническом задании.

В приложении реализованы следующие функции:

* Подбор рейсов с учетом пожеланий пассажиров: просмотр доступных рейсов с детализацией информации (номер рейса, дата отправления/прибытия, количество мест).
* Управление данными о перевозчиках, водителях, билетах, расписании, маршрутах, пунктах назначения.
* Выгрузка данных о таблицах в TXT.

На данный момент система готова к внедрению в эксплуатацию и способна значительно повысить эффективность работы городского автовокзала за счет сокращения рутинных операций и минимизации ошибок.

В дальнейшем планируется добавить расчет стоимости билета относительно времени покупки, наличии багажа и животных, выгрузку отчета о продажах, контроль покупки и бронирования билета относительно даты отъезда автобуса.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Алдунин, А. C#. Полное руководство / А. Алдунин. – М.: БХВ-Петербург, 2022. – 1104 с.
2. Hock Training [Электронный ресурс]. Режим доступа: https://www.hocktraining.com/blog/preimuschestva-yazyka-python?ysclid=m9e2o81o3e742355395.
3. Орлов, А. Visual Studio 2022. Полное руководство / А. Орлов. – М.: БХВ-Петербург, 2022. – 896 с.
4. Руссомано, Д. Microsoft SQL Server 2022. Полное руководство / Д. Руссомано. – М.: БХВ-Петербург, 2020. – 1024 с.
5. Фриден, К. Проектирование пользовательского интерфейса / К. Фриден. – М.: Питер, 2021. – 432 с.
6. Мартин, Р. Чистая Архитектура. Искусство разработки программного обеспечения / Р. Мартин. – СПб.: Питер, 2019. – 350 с.
7. Microsoft Developer Network (MSDN) [Электронный ресурс]. – Режим доступа: <https://docs.microsoft.com/ru-ru/dotnet/csharp/>.

Приложение А

(справочное)

**Результат работы программы**

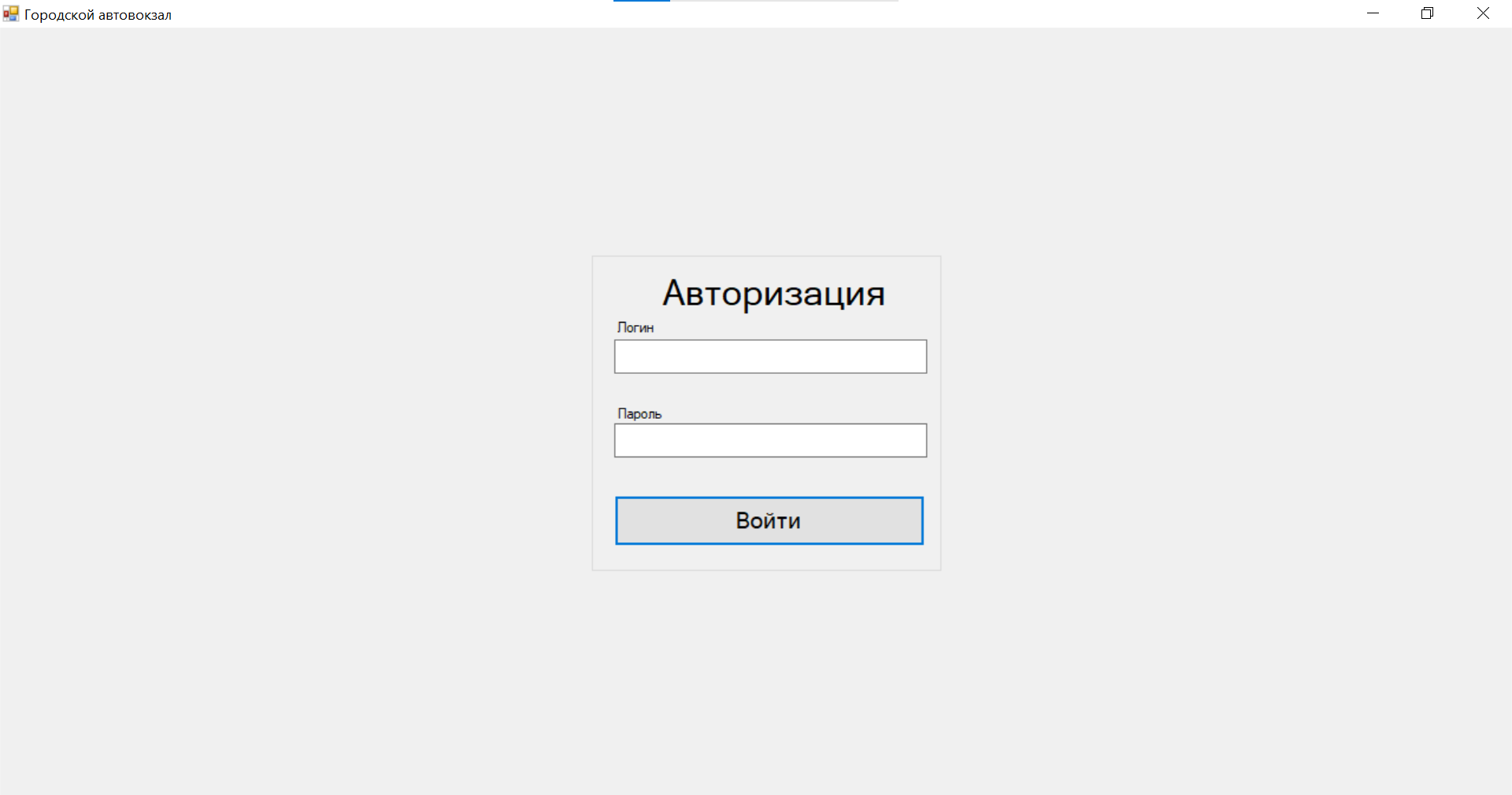
****

Рисунок – Запуск программы

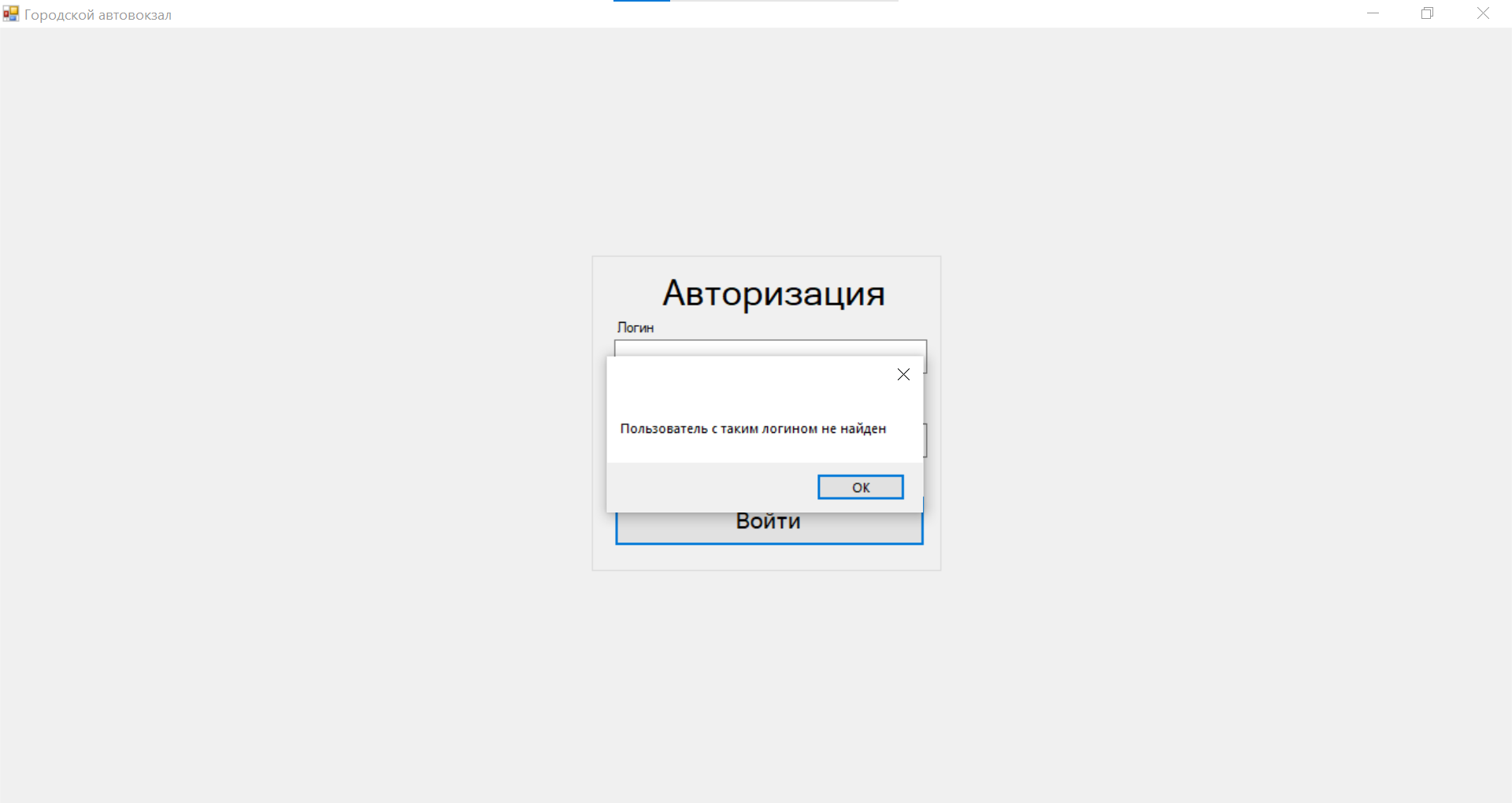


Рисунок – Предупреждение, если не введен логин или он неверен

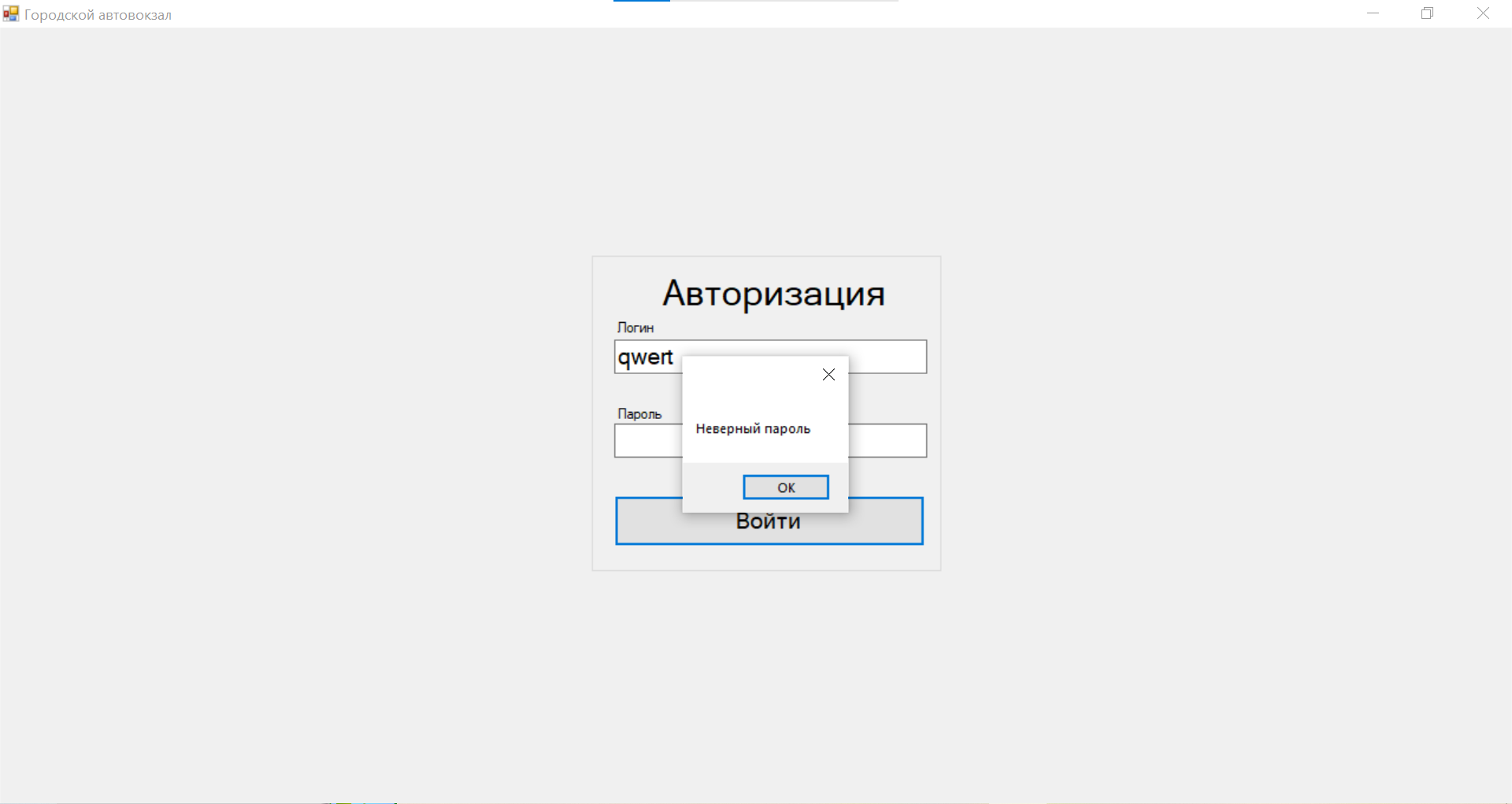


Рисунок – Предупреждение, если не введен пароль или он неверен

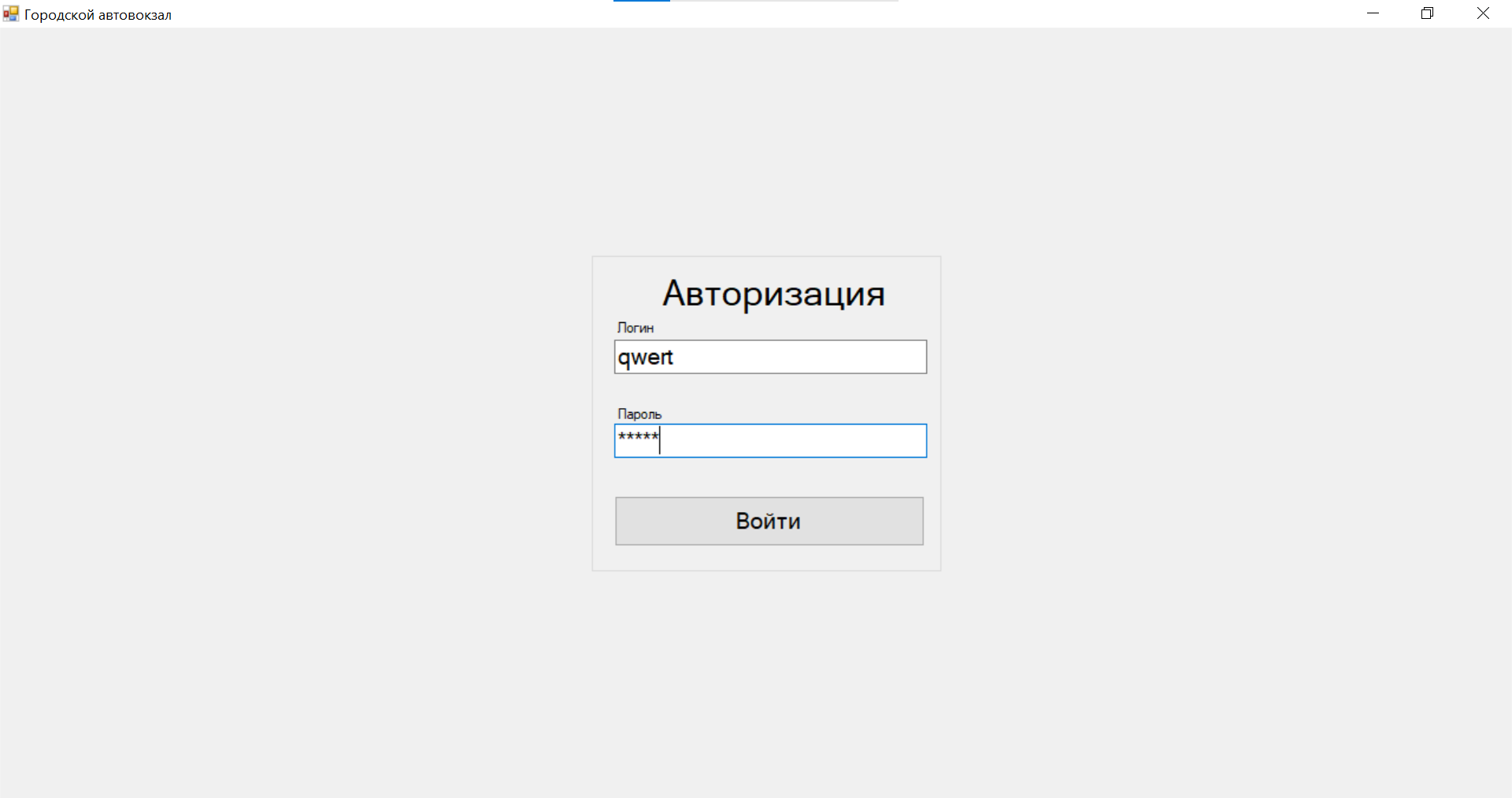


Рисунок – Ввод данных в поля авторизации

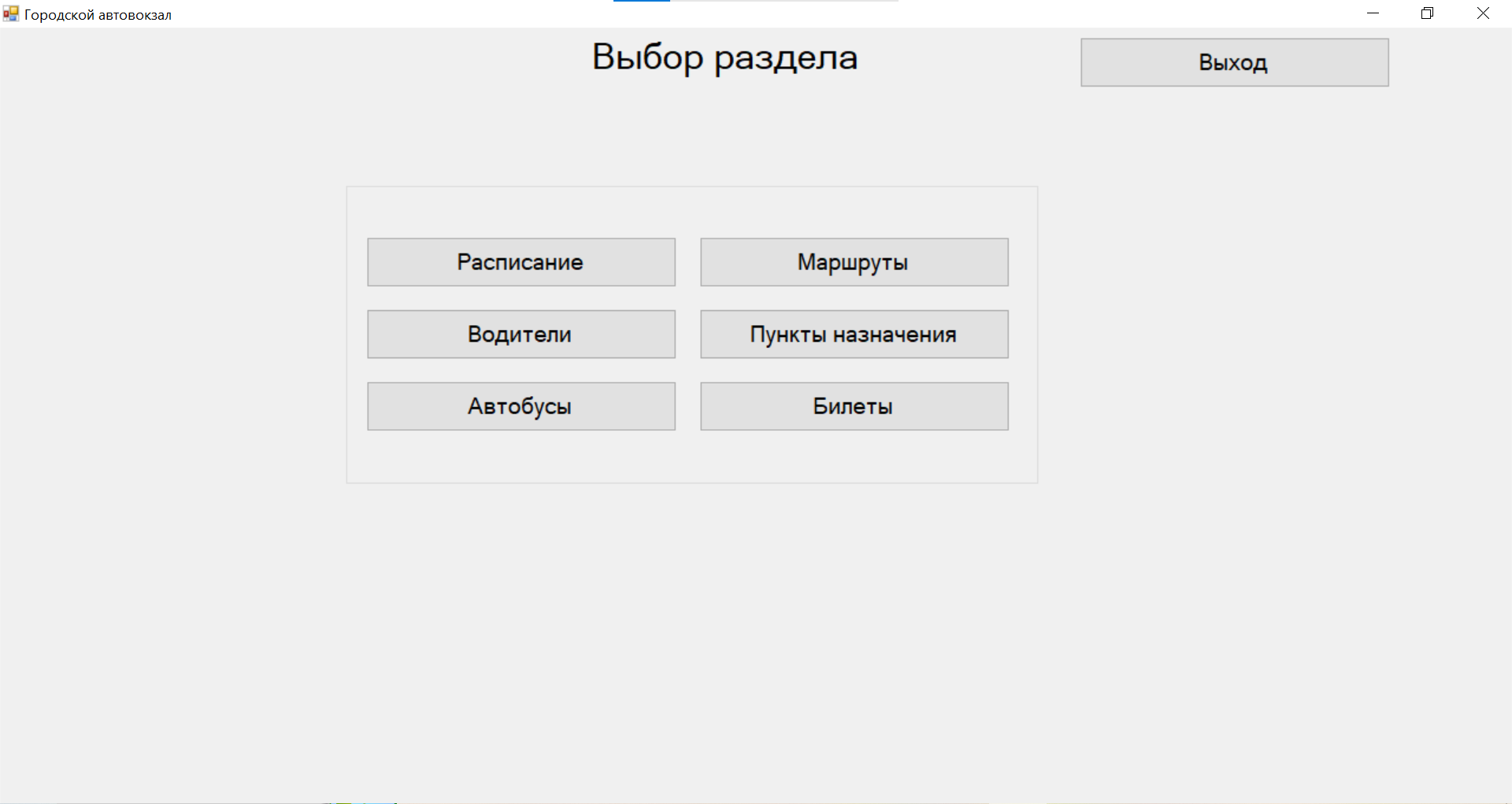


Рисунок – Главный раздел

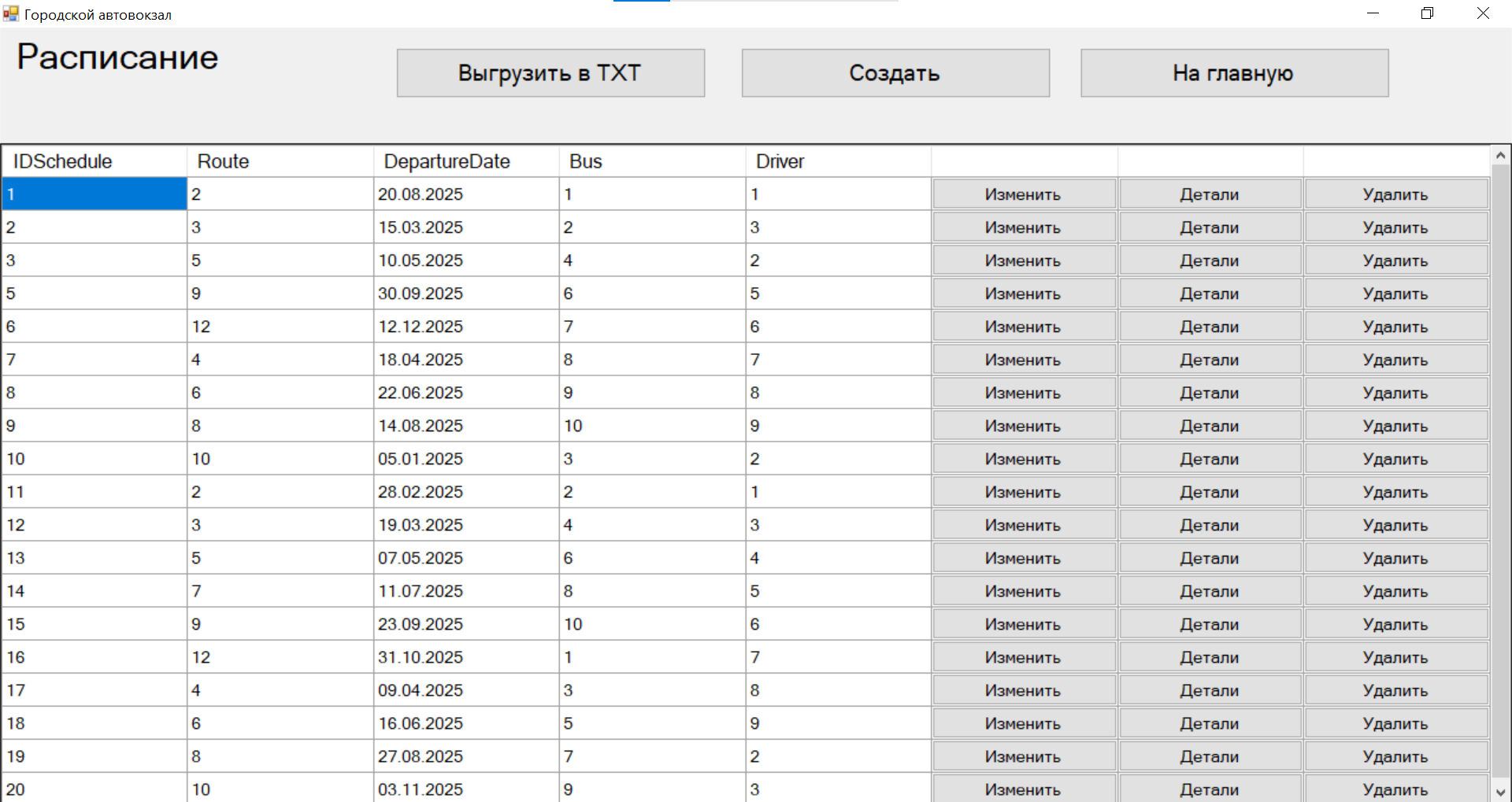


Рисунок – Раздел Расписание

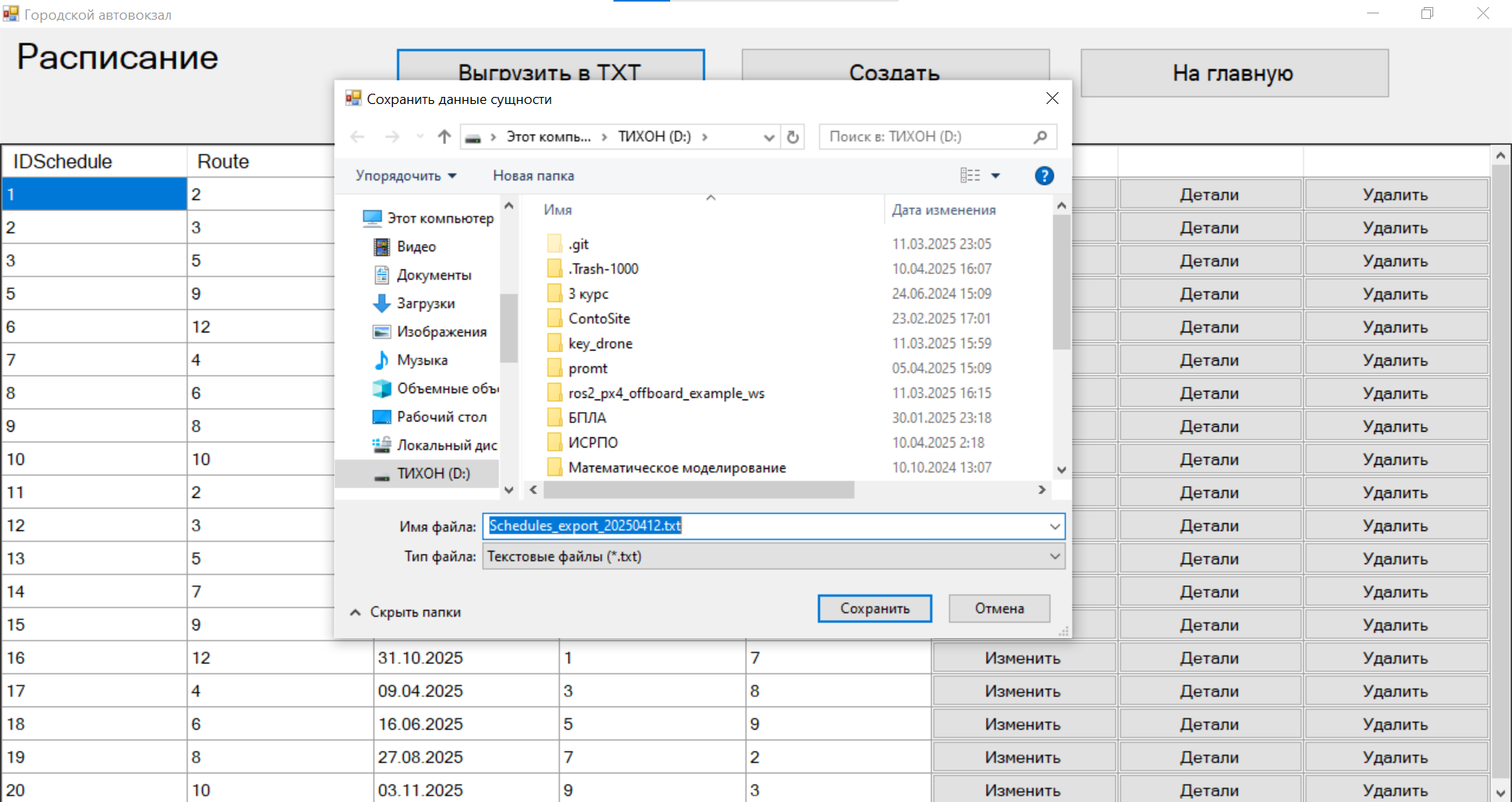


Рисунок – Выбор выгрузки данных таблицы в TXT

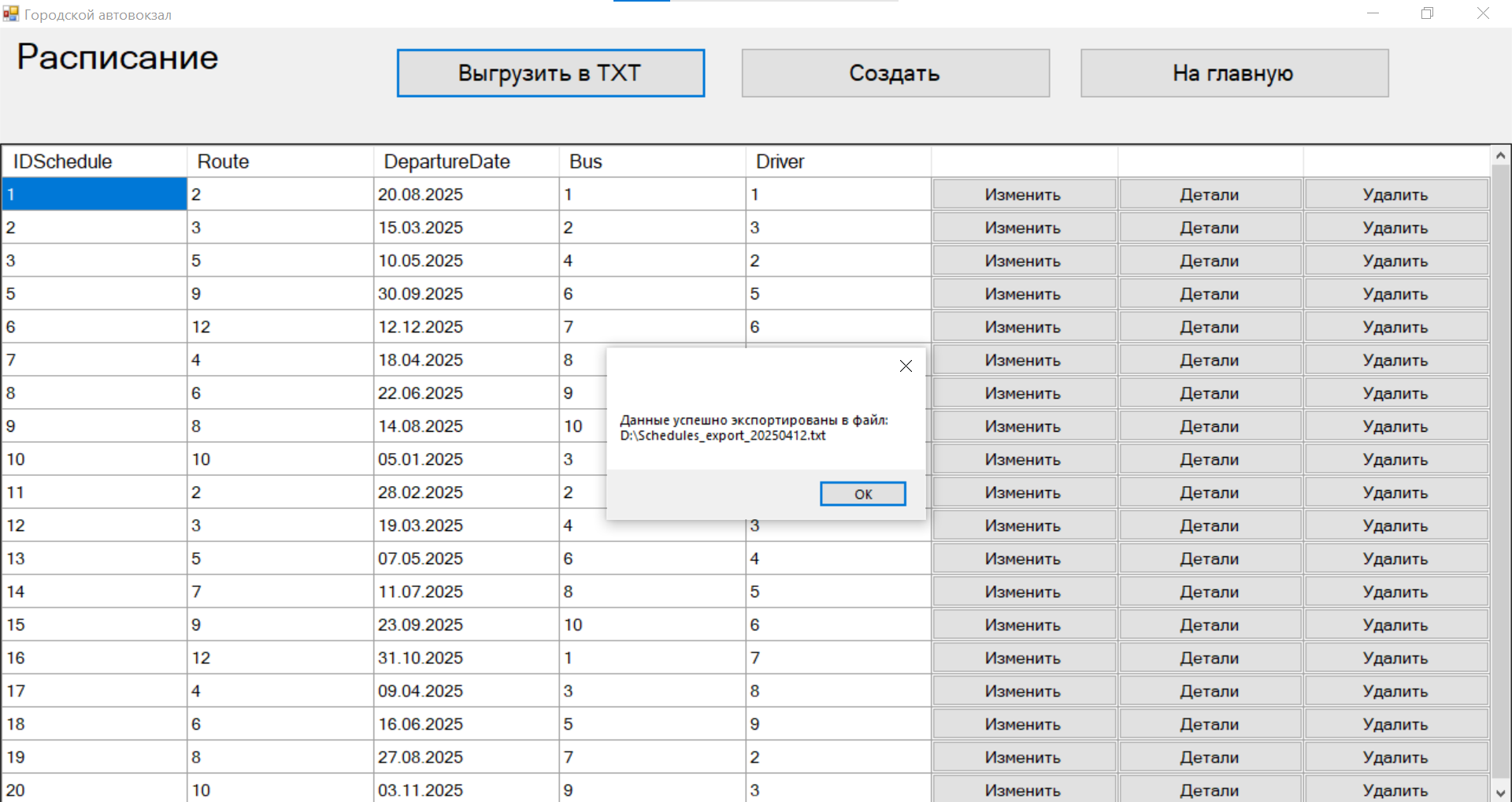


Рисунок – Результат выгрузки данных

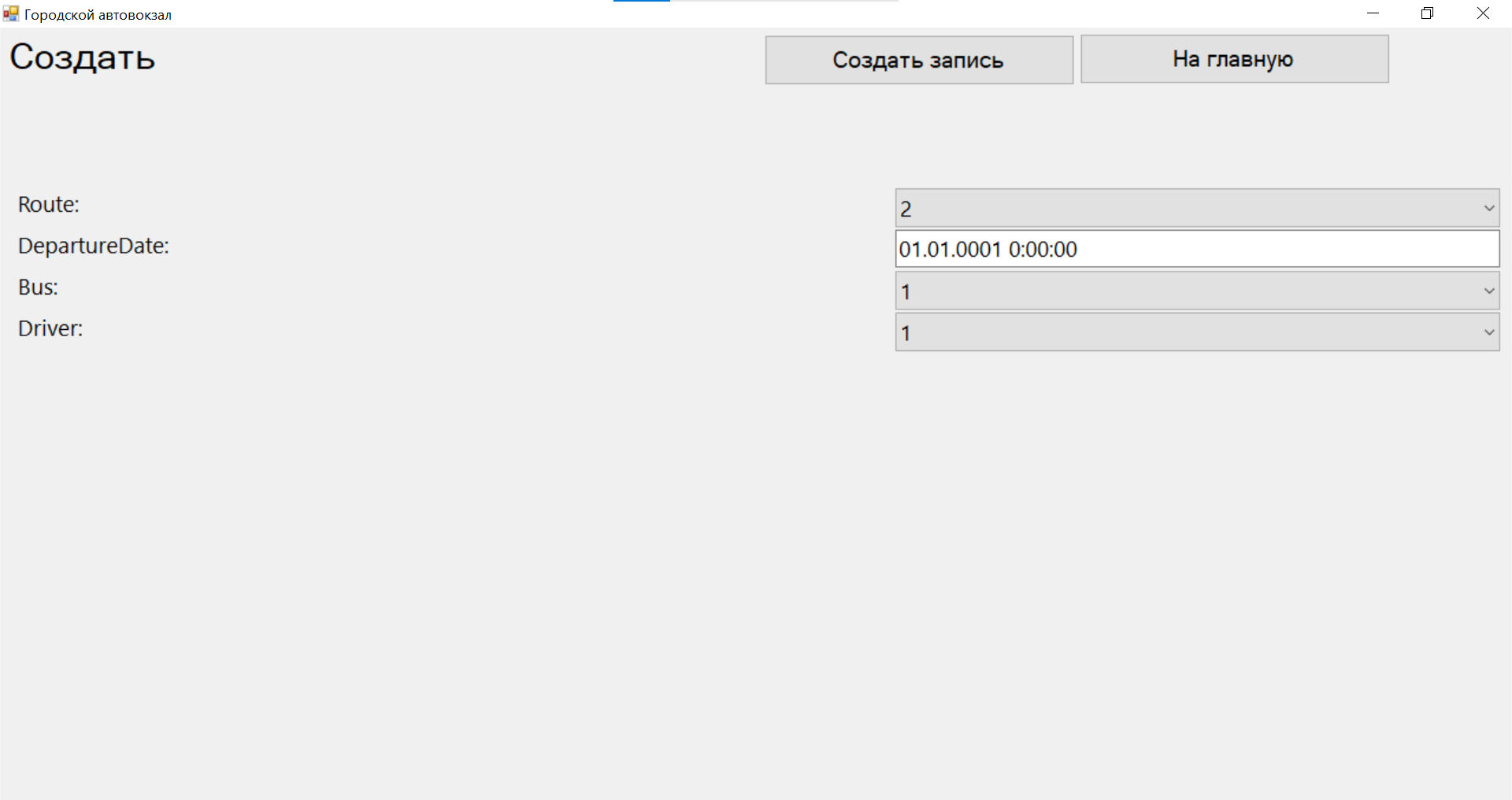


Рисунок – Раздел создания записи

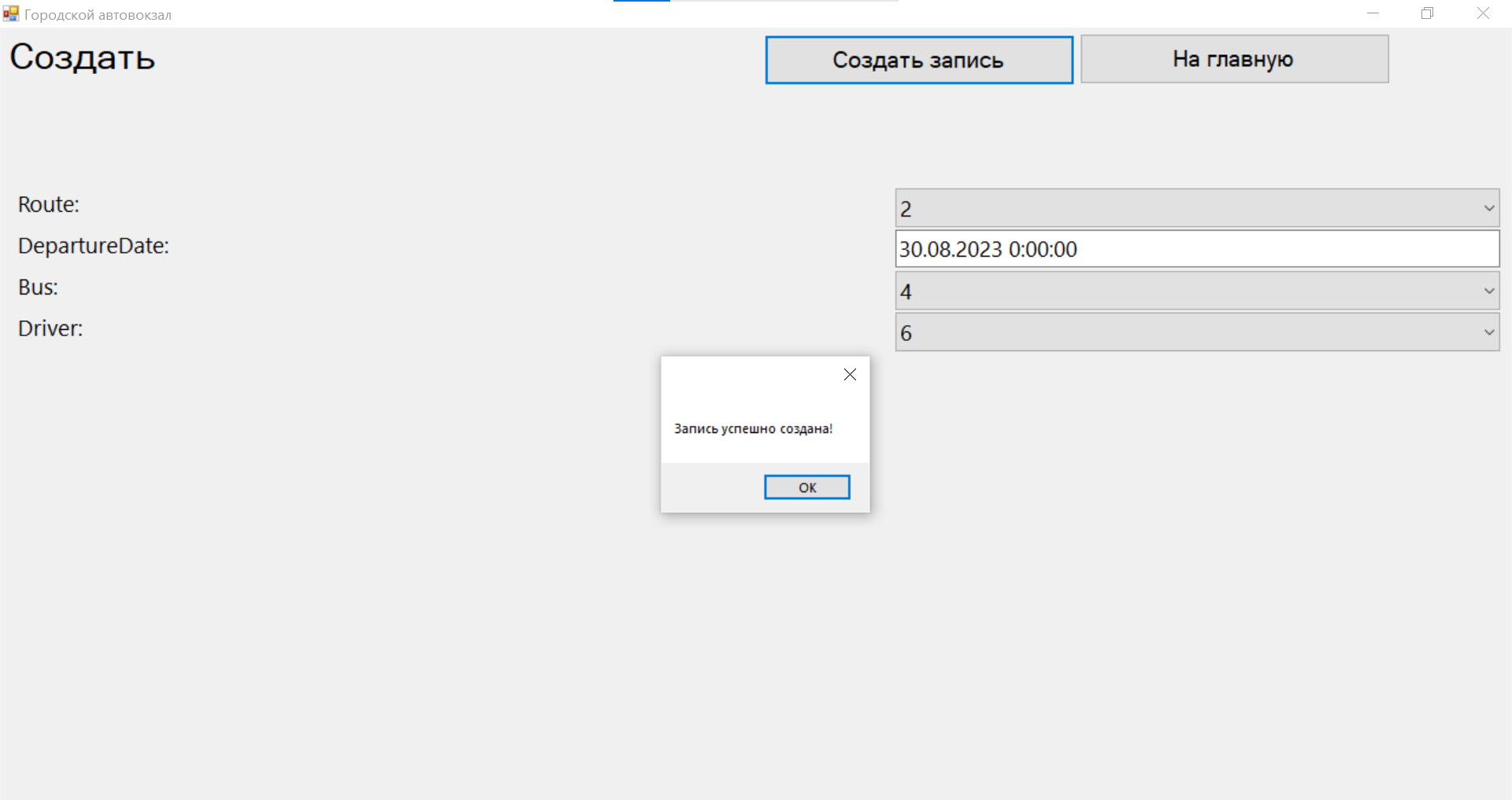


Рисунок – Результат создания записи

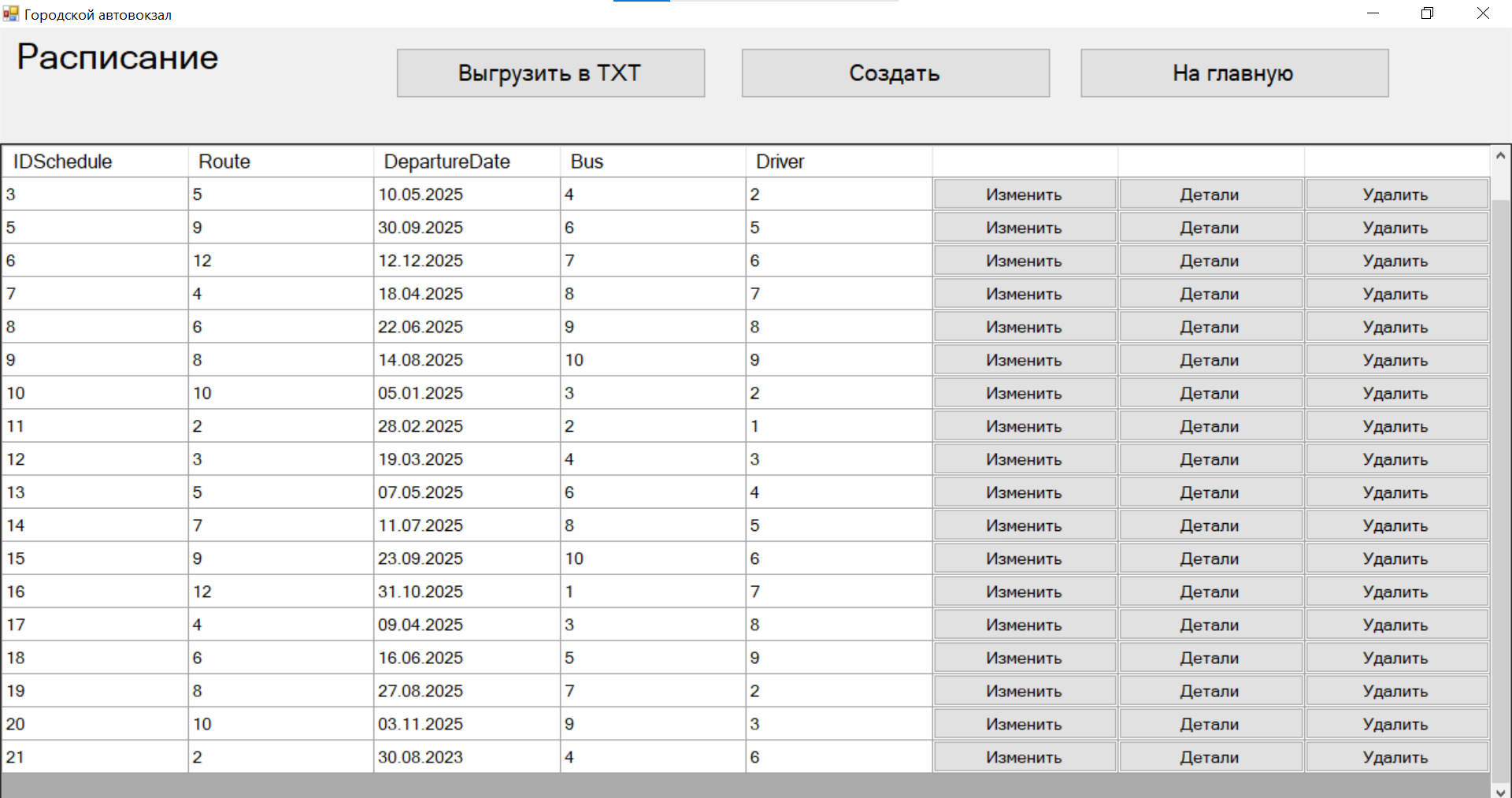


Рисунок – Обновленная таблица расписания после добавления записи

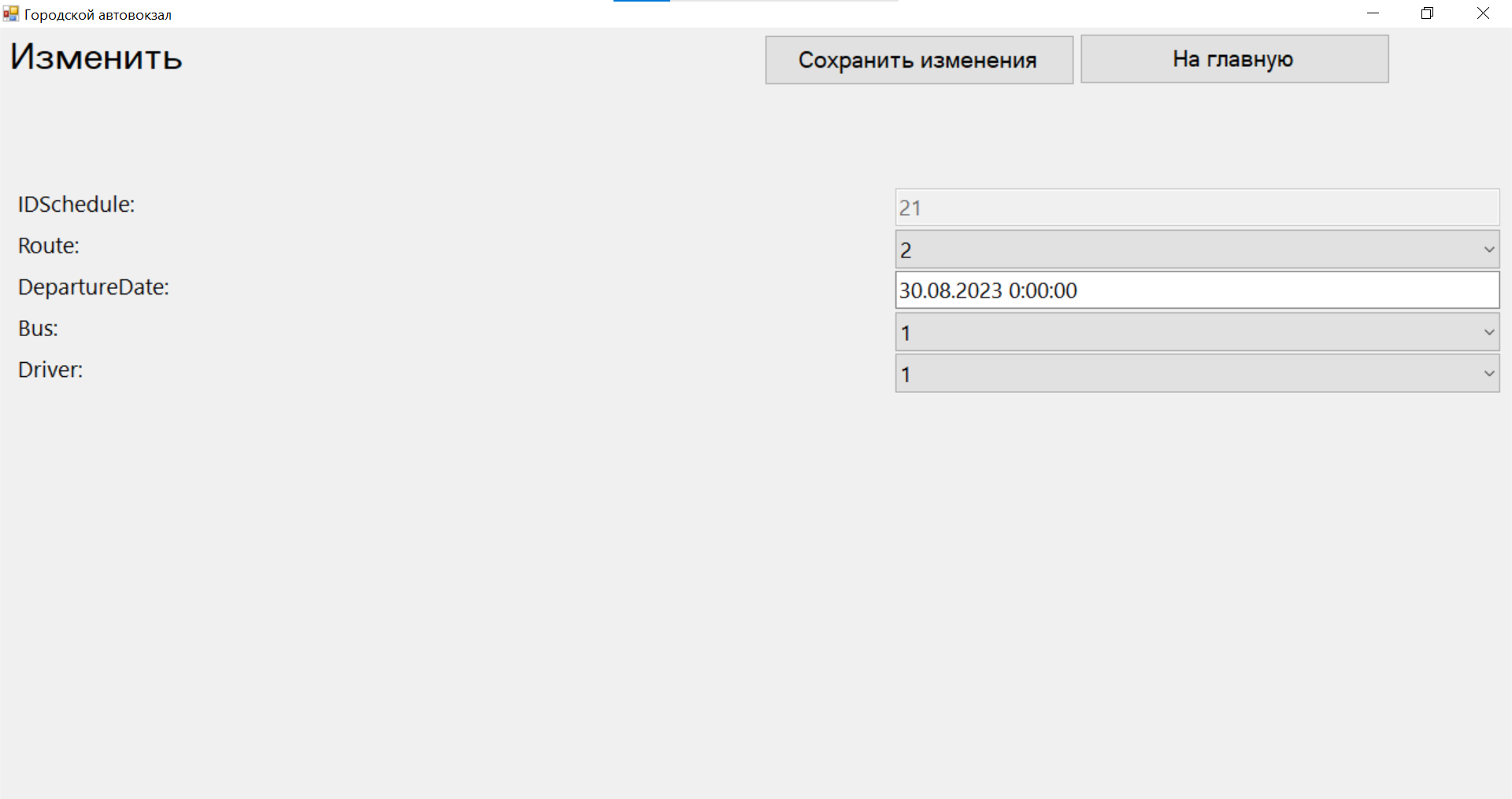


Рисунок – Изменение данных записи

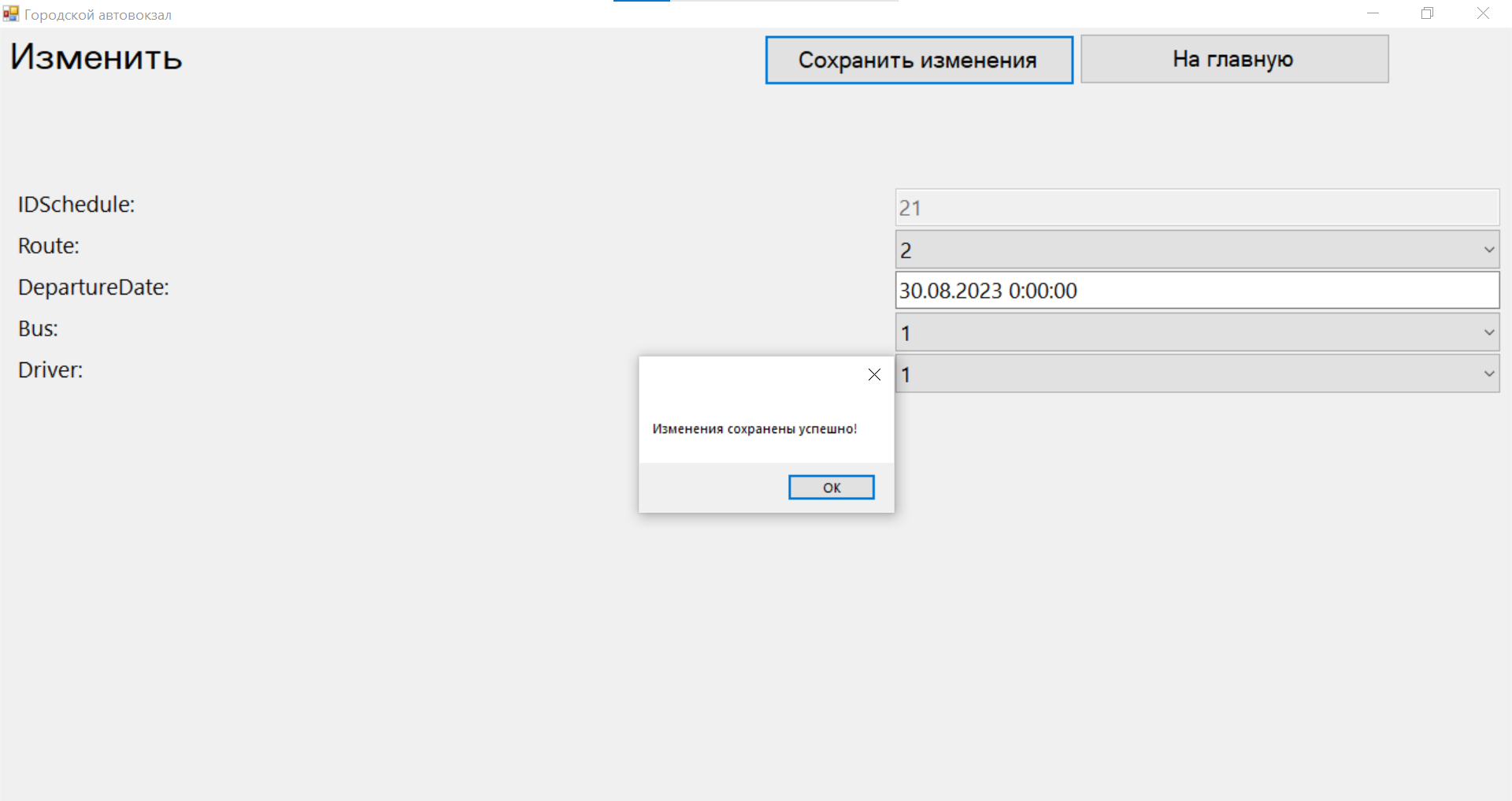


Рисунок – результат изменения записи

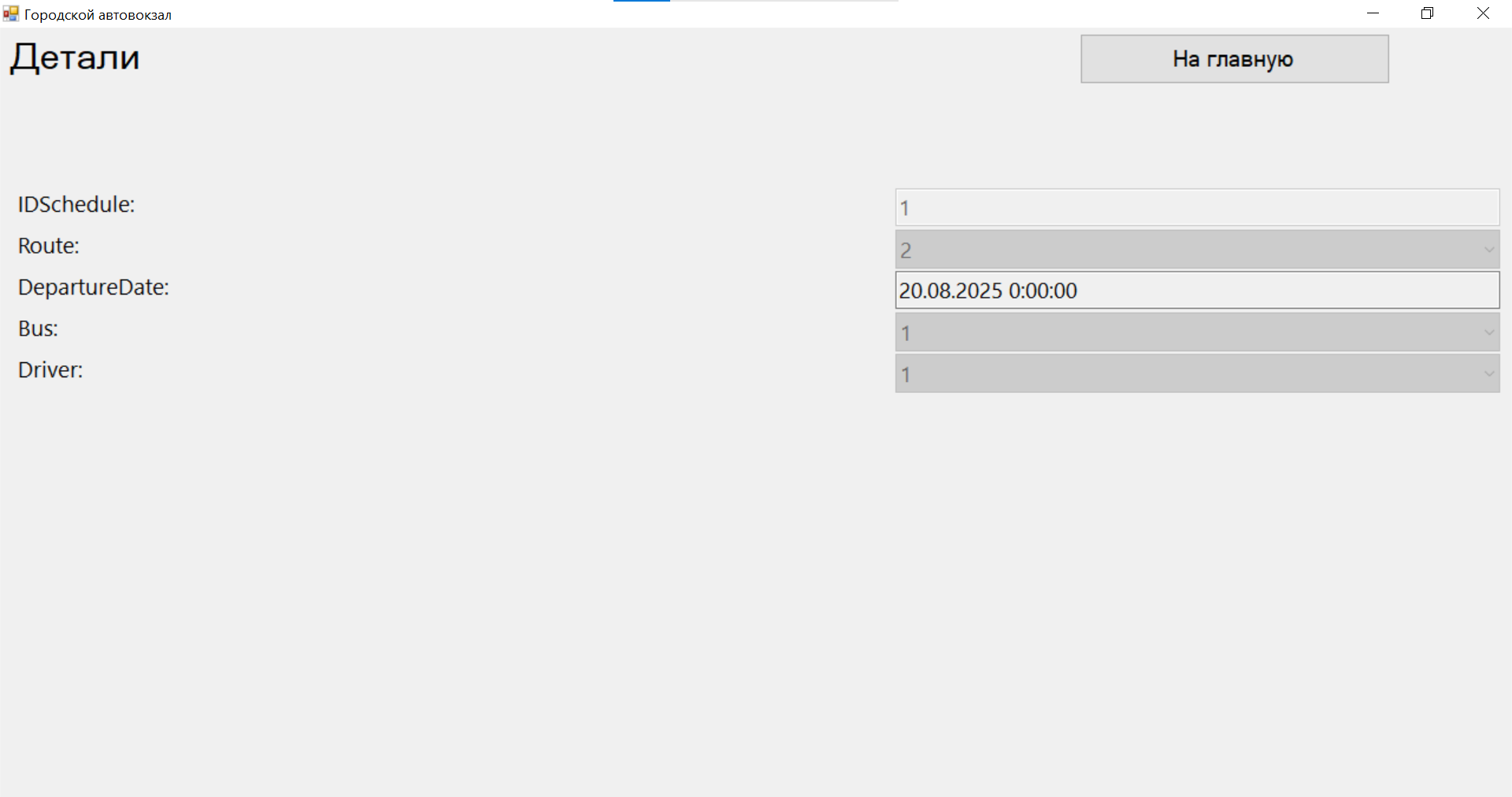


Рисунок – Раздел Детали записи

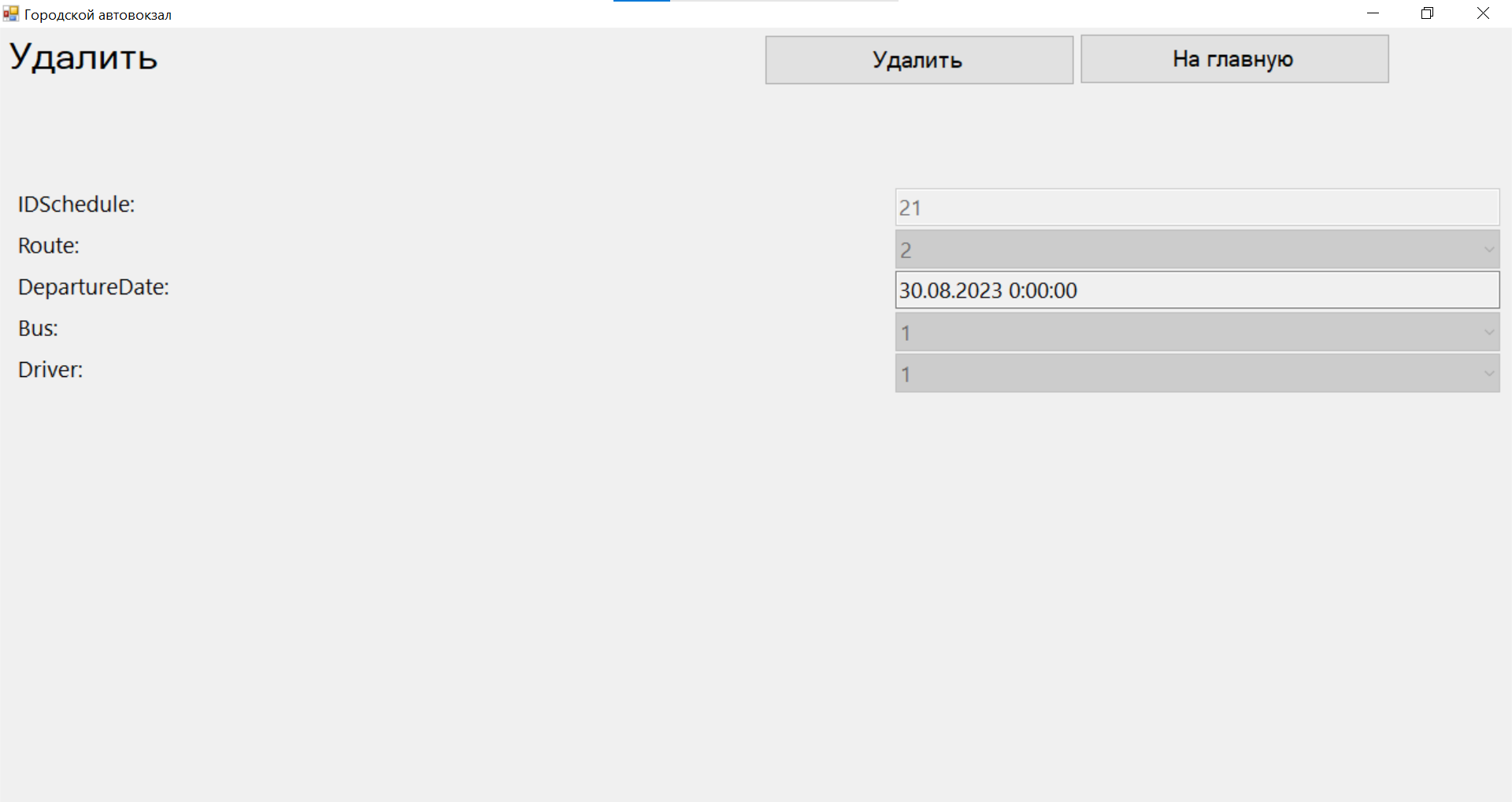


Рисунок – Удаление записи

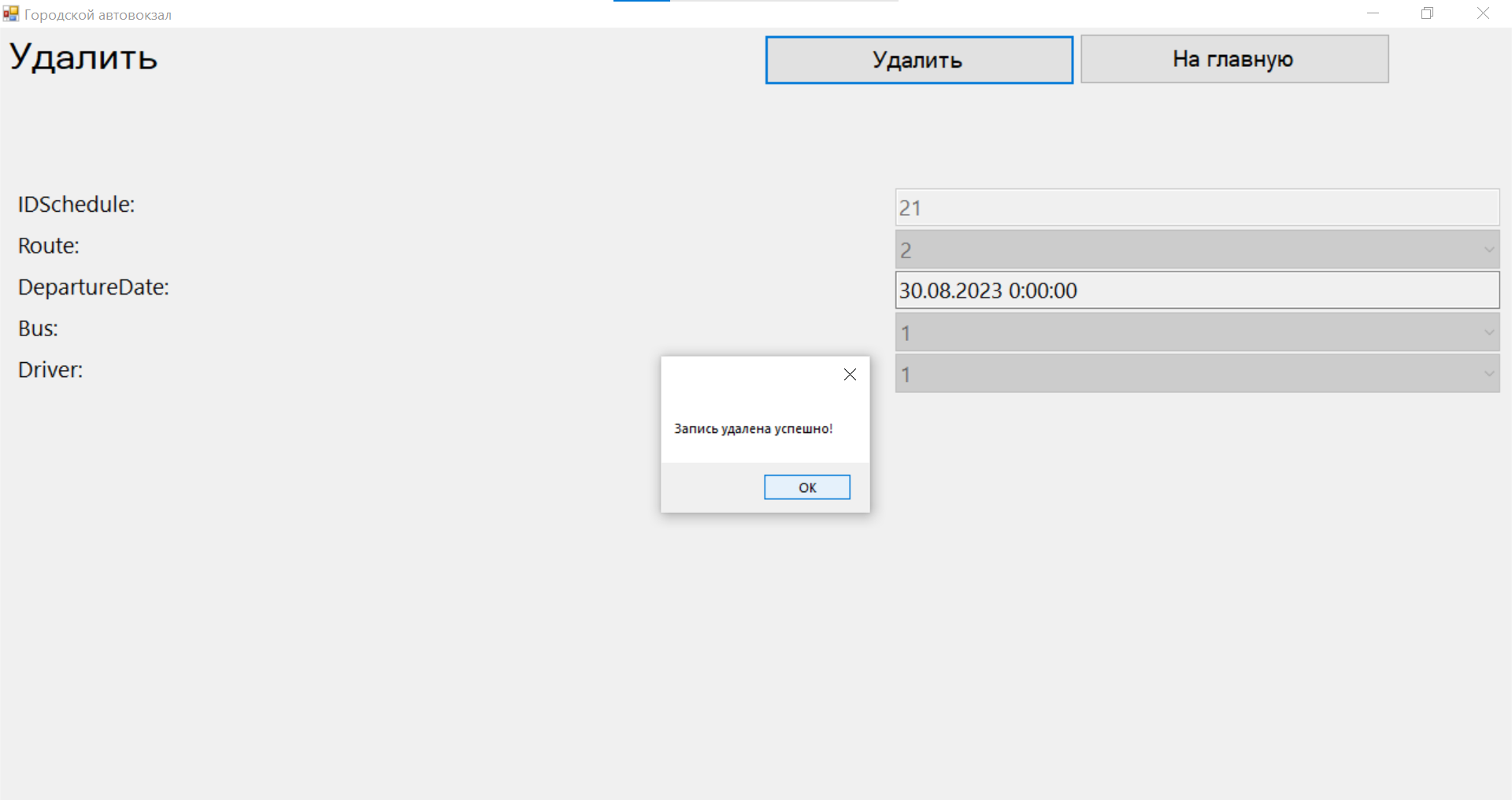


Рисунок – Результат удаления записи

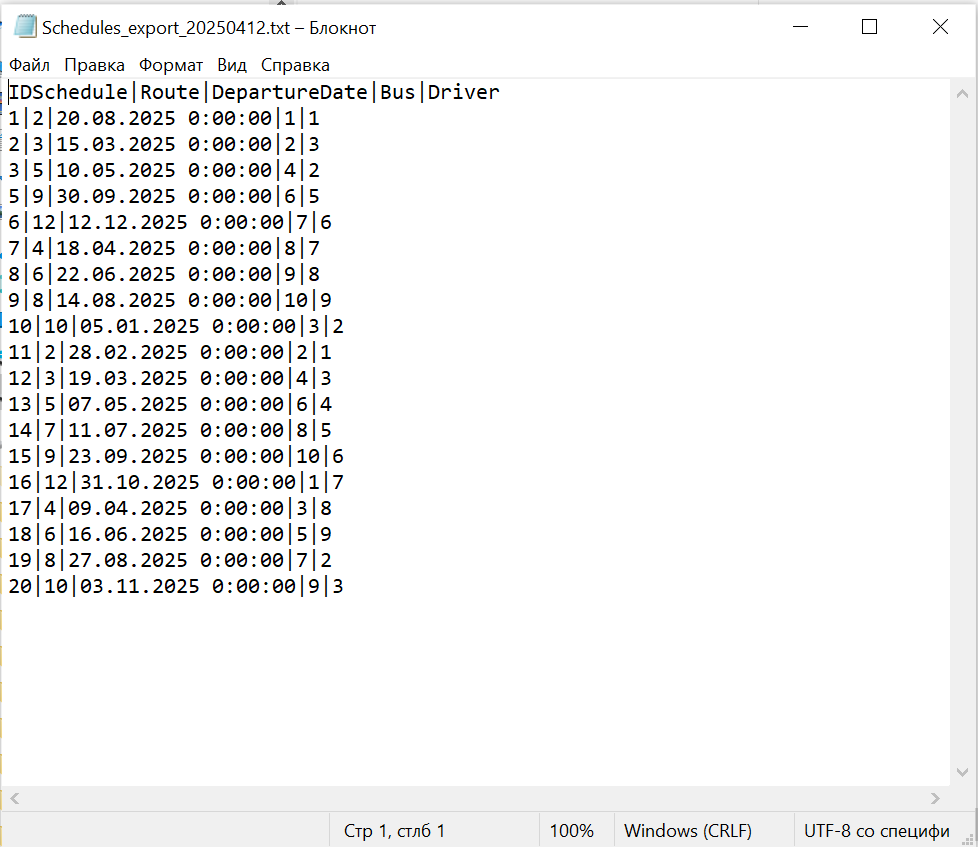


Рисунок – Результат выгрузки данных из таблицы

**Диаграмма класса**

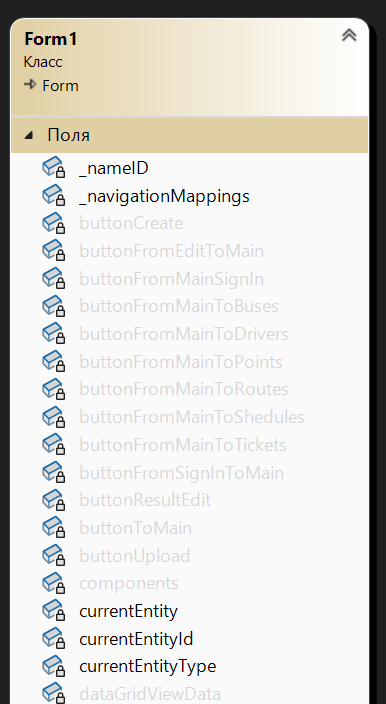


Рисунок – Диаграмма класса Form1

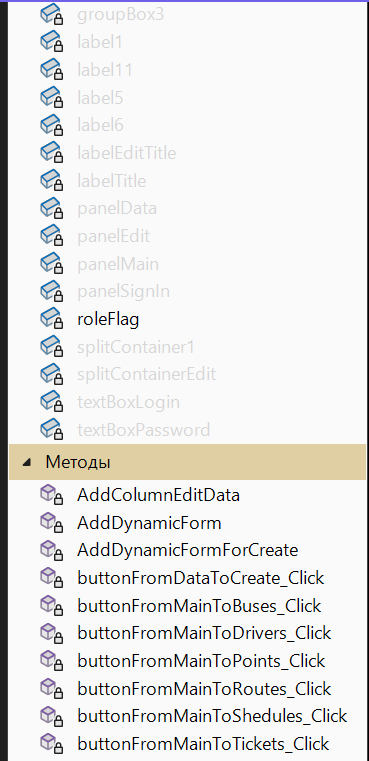


Рисунок – Диаграмма класса Form1

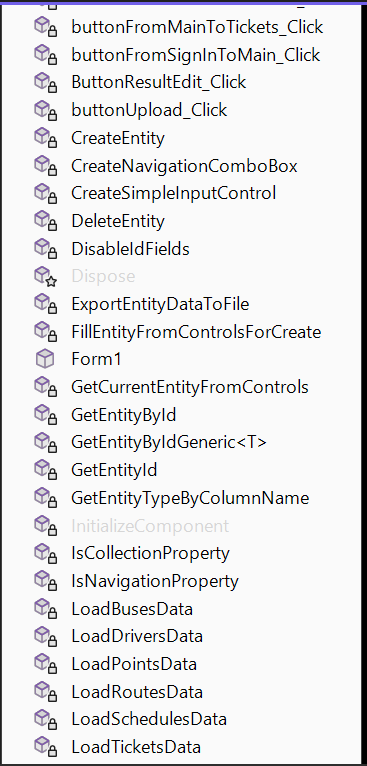


Рисунок – Диаграмма класса Form1

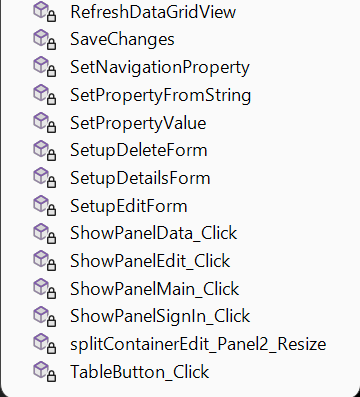


Рисунок – Диаграмма класса Form1

Приложение Б

(справочное)

**Исходный код приложения**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.Entity;

using System.Drawing;

using System.Linq;

using System.Reflection;

using System.Runtime.Remoting.Contexts;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using WindowsFormsAppcCityBusStation.Models;

using Newtonsoft.Json;

using System.Data.Entity.Infrastructure;

using System.Diagnostics;

using System.Data.Common;

using System.IO;

namespace WindowsFormsAppcCityBusStation

{

public partial class Form1 : Form

{

private dynamic currentEntity;

private Type currentEntityType;

private int currentEntityId;

private bool roleFlag;

private readonly Dictionary<string, string> \_navigationMappings = new Dictionary<string, string>

{

{ "Carrier", "Carriers" },

{ "PlaceOfDeparture", "Destinations" },

{ "PlaceOfArrival", "Destinations" },

{ "Driver", "Drivers" },

{ "Bus", "Buses" },

{ "Route", "Routes" },

{ "Schedule", "Schedules" }

};

List<string> \_nameID = new List<string>() { "IDBus", "IDCarrier", "IDDestination", "IDDriver", "IDRoute", "IDSchedule", "IDTicket" };

public Form1()

{

InitializeComponent();

buttonToMain.Click += ShowPanelMain\_Click;

buttonFromSignInToMain.Click += buttonFromSignInToMain\_Click;

dataGridViewData.CellContentClick += TableButton\_Click;

buttonFromEditToMain.Click += ShowPanelMain\_Click;

splitContainerEdit.Panel2.Resize += splitContainerEdit\_Panel2\_Resize;

buttonCreate.Click += buttonFromDataToCreate\_Click;

buttonResultEdit.Click += ButtonResultEdit\_Click;

buttonUpload.Click += buttonUpload\_Click;

buttonFromMainSignIn.Click += ShowPanelSignIn\_Click;

}

private void ShowPanelData\_Click()

{

panelData.Visible = true;

panelMain.Visible = false;

panelSignIn.Visible = false;

panelEdit.Visible = false;

}

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

{

panelMain.Visible = true;

panelSignIn.Visible = false;

panelEdit.Visible = false;

panelData.Visible = false;

}

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

{

panelSignIn.Visible = true;

panelMain.Visible = false;

panelData.Visible = false;

panelEdit.Visible = false;

textBoxLogin.Text = "";

textBoxPassword.Text = "";

}

private void ShowPanelEdit\_Click()

{

panelEdit.Visible = true;

panelSignIn.Visible = false;

panelMain.Visible = false;

panelData.Visible = false;

}

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

{

LoadSchedulesData();

}

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

{

LoadRoutesData();

}

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

{

LoadDriversData();

}

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

{

LoadPointsData();

}

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

{

LoadBusesData();

}

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

{

LoadTicketsData();

}

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

{

var idColumn = dataGridViewData.Columns

.Cast<DataGridViewColumn>()

.FirstOrDefault(c => c.Name.StartsWith("ID"));

var entityType = GetEntityTypeByColumnName(idColumn.Name);

currentEntityType = entityType;

dynamic newEntity = Activator.CreateInstance(entityType);

buttonResultEdit.Visible = true;

labelEditTitle.Text = "Создать";

buttonResultEdit.Text = "Создать запись";

AddDynamicFormForCreate(newEntity);

ShowPanelEdit\_Click();

}

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

{

try

{

var idColumn = dataGridViewData.Columns

.Cast<DataGridViewColumn>()

.FirstOrDefault(c => c.Name.StartsWith("ID"));

if (idColumn == null)

{

MessageBox.Show("Не найден столбец с ID");

return;

}

var entityType = GetEntityTypeByColumnName(idColumn.Name);

if (entityType == null)

{

MessageBox.Show("Не удалось определить тип сущности");

return;

}

var saveFileDialog = new SaveFileDialog

{

Filter = "Текстовые файлы (\*.txt)|\*.txt",

Title = "Сохранить данные сущности",

FileName = $"{entityType.Name}\_export\_{DateTime.Now:yyyyMMdd}.txt"

};

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

return;

ExportEntityDataToFile(entityType, saveFileDialog.FileName);

MessageBox.Show($"Данные успешно экспортированы в файл:\n{saveFileDialog.FileName}");

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при экспорте данных: {ex.Message}");

Console.WriteLine($"Ошибка при экспорте данных: {ex.Message}");

}

}

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

{

string login = textBoxLogin.Text;

string password = textBoxPassword.Text;

using (var context = new CityBusStationEntities())

{

try

{

var user = context.Autorization.FirstOrDefault(a => a.Login == login);

if (user == null)

{

MessageBox.Show("Пользователь с таким логином не найден");

return;

}

if (user.Password != password)

{

MessageBox.Show("Неверный пароль");

return;

}

int userRole = user.Role;

var role = context.Roles.Find(userRole);

if (role == null)

{

MessageBox.Show("Такой роли нет!");

return;

}

string roleName = role.Role;

if (role == null)

{

MessageBox.Show("Такой роли нет!");

return;

}

if (roleName == "Admin")

{

roleFlag = true;

}

if (roleName == "Cashier")

{

roleFlag = false;

}

ShowPanelMain\_Click(null, null);

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка авторизации: {ex.Message}");

}

}

}

private void ExportEntityDataToFile(Type entityType, string filePath)

{

using (var context = new CityBusStationEntities())

{

try

{

var setMethod = typeof(DbContext).GetMethods()

.FirstOrDefault(m => m.Name == "Set" &&

m.IsGenericMethod &&

m.GetParameters().Length == 0);

if (setMethod == null)

throw new InvalidOperationException("Не удалось найти метод Set<T>");

var genericSetMethod = setMethod.MakeGenericMethod(entityType);

dynamic dbSet = genericSetMethod.Invoke(context, null);

var data = new List<object>();

foreach (var item in dbSet)

{

data.Add(item);

}

var properties = entityType.GetProperties()

.Where(p => !IsNavigationProperty(p) &&

!IsCollectionProperty(p) &&

(p.PropertyType.IsPrimitive ||

p.PropertyType == typeof(string) ||

p.PropertyType == typeof(DateTime)))

.ToList();

var sb = new StringBuilder();

sb.AppendLine(string.Join("|", properties.Select(p => p.Name)));

foreach (var item in data)

{

var values = properties.Select(p =>

{

var value = p.GetValue(item);

return value != null ? value.ToString() : "NULL";

});

sb.AppendLine(string.Join("|", values));

}

File.WriteAllText(filePath, sb.ToString(), Encoding.UTF8);

}

catch (Exception ex)

{

throw new Exception($"Ошибка при экспорте данных: {ex.Message}", ex);

}

}

}

private void LoadSchedulesData()

{

labelTitle.Text = "Расписание";

using (var context = new CityBusStationEntities())

{

var schedules = context.Schedules.Select(s => new

{

s.IDSchedule,

s.Route,

s.DepartureDate,

s.Bus,

s.Driver

}).ToList();

dataGridViewData.DataSource = schedules;

}

if (roleFlag)

AddColumnEditData(dataGridViewData);

else

{

buttonCreate.Visible = false;

if (dataGridViewData.Columns.Contains("Edit")) dataGridViewData.Columns.Remove("Edit");

if (dataGridViewData.Columns.Contains("Delete")) dataGridViewData.Columns.Remove("Delete");

if (dataGridViewData.Columns.Contains("Details")) dataGridViewData.Columns.Remove("Details");

}

ShowPanelData\_Click();

}

private void LoadRoutesData()

{

try

{

labelTitle.Text = "Маршруты";

using (var context = new CityBusStationEntities())

{

var routes = context.Routes.Select(r => new

{

r.IDRoute,

r.PlaceOfDeparture,

r.PlaceOfArrival,

r.TimeOfDeparture,

r.TravelTimeInHours

}).ToList();

dataGridViewData.DataSource = routes;

}

if (roleFlag)

AddColumnEditData(dataGridViewData);

else

{

buttonCreate.Visible = false;

if (dataGridViewData.Columns.Contains("Edit")) dataGridViewData.Columns.Remove("Edit");

if (dataGridViewData.Columns.Contains("Delete")) dataGridViewData.Columns.Remove("Delete");

if (dataGridViewData.Columns.Contains("Details")) dataGridViewData.Columns.Remove("Details");

}

ShowPanelData\_Click();

}

catch (Exception ex)

{

Console.WriteLine(ex.ToString());

}

}

private void LoadDriversData()

{

labelTitle.Text = "Водители";

using (var context = new CityBusStationEntities())

{

var drivers = context.Drivers.Select(d => new

{

d.IDDriver,

d.Surname,

d.Name,

d.Patronymic,

d.Age,

d.YearsOfDrivingExperience

}).ToList();

dataGridViewData.DataSource = drivers;

}

if (roleFlag)

AddColumnEditData(dataGridViewData);

else

{

buttonCreate.Visible = false;

if (dataGridViewData.Columns.Contains("Edit")) dataGridViewData.Columns.Remove("Edit");

if (dataGridViewData.Columns.Contains("Delete")) dataGridViewData.Columns.Remove("Delete");

if (dataGridViewData.Columns.Contains("Details")) dataGridViewData.Columns.Remove("Details");

}

ShowPanelData\_Click();

}

private void LoadPointsData()

{

labelTitle.Text = "Пункты назначения";

using (var context = new CityBusStationEntities())

{

var destinations = context.Destinations.Select(d => new

{

d.IDDestination,

d.City,

d.Street,

d.House

}).ToList();

dataGridViewData.DataSource = destinations;

}

if (roleFlag)

AddColumnEditData(dataGridViewData);

else

{

buttonCreate.Visible = false;

if (dataGridViewData.Columns.Contains("Edit")) dataGridViewData.Columns.Remove("Edit");

if (dataGridViewData.Columns.Contains("Delete")) dataGridViewData.Columns.Remove("Delete");

if (dataGridViewData.Columns.Contains("Details")) dataGridViewData.Columns.Remove("Details");

}

ShowPanelData\_Click();

}

private void LoadBusesData()

{

labelTitle.Text = "Автобусы";

using (var context = new CityBusStationEntities())

{

var buses = context.Buses.Select(b => new

{

b.IDBus,

b.Carrier,

b.BusNumber,

b.NumberOfSeats,

b.FireExtinguisher

}).ToList();

dataGridViewData.DataSource = buses;

}

if (roleFlag)

AddColumnEditData(dataGridViewData);

else

{

buttonCreate.Visible = false;

if (dataGridViewData.Columns.Contains("Edit")) dataGridViewData.Columns.Remove("Edit");

if (dataGridViewData.Columns.Contains("Delete")) dataGridViewData.Columns.Remove("Delete");

if (dataGridViewData.Columns.Contains("Details")) dataGridViewData.Columns.Remove("Details");

}

ShowPanelData\_Click();

}

private void LoadTicketsData()

{

labelTitle.Text = "Билеты";

buttonCreate.Visible = true;

using (var context = new CityBusStationEntities())

{

var tickets = context.Tickets.Select(t => new

{

t.IDTicket,

t.Surname,

t.Name,

t.Patronymic,

t.Age,

t.PassportSeriesAndNumber,

t.Schedule,

t.PlaceNumber,

t.Pets,

t.BoughtOut

}).ToList();

dataGridViewData.DataSource = tickets;

}

AddColumnEditData(dataGridViewData);

ShowPanelData\_Click();

}

private void AddColumnEditData(DataGridView d)

{

if (d.Columns.Contains("Edit")) d.Columns.Remove("Edit");

if (d.Columns.Contains("Delete")) d.Columns.Remove("Delete");

if (d.Columns.Contains("Details")) d.Columns.Remove("Details");

var boolColumns = new List<string>();

foreach (DataGridViewColumn column in d.Columns)

{

if (column.ValueType == typeof(bool?) || column.ValueType == typeof(bool))

{

boolColumns.Add(column.Name);

}

}

foreach (var columnName in boolColumns)

{

var column = d.Columns[columnName];

var checkBoxColumn = new DataGridViewCheckBoxColumn

{

HeaderText = column.HeaderText,

Name = column.Name,

DataPropertyName = column.DataPropertyName,

Width = column.Width,

TrueValue = true,

FalseValue = false

};

int columnIndex = column.Index;

d.Columns.Remove(column);

d.Columns.Insert(columnIndex, checkBoxColumn);

}

var editButton = new DataGridViewButtonColumn();

editButton.Name = "Edit";

editButton.HeaderText = "";

editButton.Text = "Изменить";

editButton.UseColumnTextForButtonValue = true;

d.Columns.Add(editButton);

var detailsButton = new DataGridViewButtonColumn();

detailsButton.Name = "Details";

detailsButton.HeaderText = "";

detailsButton.Text = "Детали";

detailsButton.UseColumnTextForButtonValue = true;

d.Columns.Add(detailsButton);

var deleteButton = new DataGridViewButtonColumn();

deleteButton.Name = "Delete";

deleteButton.HeaderText = "";

deleteButton.Text = "Удалить";

deleteButton.UseColumnTextForButtonValue = true;

d.Columns.Add(deleteButton);

d.Columns["Edit"].DisplayIndex = d.Columns.Count – 3;

d.Columns["Details"].DisplayIndex = d.Columns.Count – 2;

d.Columns["Delete"].DisplayIndex = d.Columns.Count – 1;

}

private void TableButton\_Click(object sender, DataGridViewCellEventArgs e)

{

if (e.RowIndex < 0) return;

var columnName = dataGridViewData.Columns[e.ColumnIndex].Name;

var idColumn = dataGridViewData.Columns

.Cast<DataGridViewColumn>()

.FirstOrDefault(c => c.Name.StartsWith("ID"));

if (idColumn == null)

{

MessageBox.Show("Не найден столбец с ID");

return;

}

var cellValue = dataGridViewData.Rows[e.RowIndex].Cells[idColumn.Index].Value;

if (cellValue == null || !int.TryParse(cellValue.ToString(), out int idValue))

{

MessageBox.Show("Некорректное значение ID");

return;

}

var entityType = GetEntityTypeByColumnName(idColumn.Name);

if (entityType == null)

{

MessageBox.Show("Неизвестный тип данных");

return;

}

dynamic entity = GetEntityById(entityType, idValue);

if (entity == null)

{

MessageBox.Show("Данные не найдены");

return;

}

switch (columnName)

{

case "Edit":

SetupEditForm(entity, "Изменить", "Сохранить изменения");

break;

case "Details":

SetupDetailsForm(entity);

break;

case "Delete":

SetupDeleteForm(entity);

break;

}

}

private Type GetEntityTypeByColumnName(string columnName)

{

var typeMap = new Dictionary<string, Type>

{

{ "IDSchedule", typeof(Schedules) },

{ "IDBus", typeof(Buses) },

{ "IDDriver", typeof(Drivers) },

{ "IDDestination", typeof(Destinations) },

{ "IDTicket", typeof(Tickets) },

{ "IDRoute", typeof(Routes) }

};

return typeMap.TryGetValue(columnName, out var type) ? type : null;

}

private object GetEntityById(Type entityType, int id)

{

var method = this.GetType().GetMethod("GetEntityByIdGeneric",

BindingFlags.NonPublic | BindingFlags.Instance)

.MakeGenericMethod(entityType);

return method.Invoke(this, new object[] { id });

}

private T GetEntityByIdGeneric<T>(int id) where T : class

{

using (var context = new CityBusStationEntities())

{

context.Configuration.LazyLoadingEnabled = false;

var entity = context.Set<T>().Find(id);

if (entity is Schedules schedule)

{

context.Entry(schedule).Reference(s => s.Buses).Load();

context.Entry(schedule).Reference(s => s.Drivers).Load();

context.Entry(schedule).Reference(s => s.Routes).Load();

context.Entry(schedule).Collection(s => s.Tickets).Load();

}

return entity;

}

}

private void SetupEditForm(dynamic entity, string title, string buttonText)

{

currentEntity = entity;

currentEntityType = entity.GetType();

currentEntityId = GetEntityId(entity);

buttonResultEdit.Visible = true;

labelEditTitle.Text = title;

buttonResultEdit.Text = buttonText;

AddDynamicForm(entity, true);

DisableIdFields();

ShowPanelEdit\_Click();

}

private void SetupDetailsForm(dynamic entity)

{

labelEditTitle.Text = "Детали";

buttonResultEdit.Visible = false;

AddDynamicForm(entity);

DisableIdFields();

ShowPanelEdit\_Click();

}

private void SetupDeleteForm(dynamic entity)

{

currentEntity = entity;

currentEntityType = entity.GetType();

currentEntityId = GetEntityId(entity);

labelEditTitle.Text = "Удалить";

buttonResultEdit.Visible = true;

buttonResultEdit.Text = "Удалить";

AddDynamicForm(entity);

DisableIdFields();

ShowPanelEdit\_Click();

}

private int GetEntityId(dynamic entity)

{

var type = (Type)entity.GetType();

var idProp = type.GetProperties()

.FirstOrDefault(p => p.Name.StartsWith("ID"));

return (int)idProp.GetValue(entity);

}

private void AddDynamicForm(dynamic entity, bool readField = false)

{

splitContainerEdit.Panel2.Controls.Clear();

int yPos = 10;

int panelWidth = splitContainerEdit.Panel2.Width;

int controlWidth = panelWidth / 2;

using (var context = new CityBusStationEntities())

{

foreach (var prop in entity.GetType().GetProperties())

{

if (IsCollectionProperty(prop)) continue;

if (IsNavigationProperty(prop)) continue;

var lbl = new Label

{

Text = $"{prop.Name}:",

Location = new Point(10, yPos),

Width = controlWidth – 20,

Height = 30,

Font = new Font("Segoe UI", 14)

};

splitContainerEdit.Panel2.Controls.Add(lbl);

object propValue = prop.GetValue(entity);

Control inputControl;

string navigationPropertyName;

if (\_navigationMappings.TryGetValue(prop.Name, out navigationPropertyName))

{

inputControl = CreateNavigationComboBox(context, navigationPropertyName, propValue, controlWidth, yPos, readField);

}

else

{

inputControl = CreateSimpleInputControl(prop, propValue, controlWidth, yPos, readField);

}

if (inputControl != null)

{

splitContainerEdit.Panel2.Controls.Add(inputControl);

yPos += 35;

}

}

}

}

private ComboBox CreateNavigationComboBox(CityBusStationEntities context, string entityTypeName, object currentValue, int xPos, int yPos, bool readField)

{

int controlWidth = (int)(splitContainerEdit.Panel2.Width \* 0.4);

int labelWidth = splitContainerEdit.Panel2.Width – controlWidth – 30;

var combo = new ComboBox

{

DropDownStyle = ComboBoxStyle.DropDownList,

Location = new Point(labelWidth + 20, yPos),

Width = controlWidth,

Height = 50,

Font = new Font("Segoe UI", 14),

DropDownHeight = 200,

Enabled = readField

};

var entityType = Assembly.GetExecutingAssembly()

.GetTypes()

.FirstOrDefault(t => t.Name == entityTypeName);

if (entityType != null)

{

try

{

var setMethod = typeof(DbContext).GetMethod("Set", Type.EmptyTypes)

.MakeGenericMethod(entityType);

var dbSet = (IQueryable)setMethod.Invoke(context, null);

var idProperty = entityType.GetProperties()

.FirstOrDefault(p => \_nameID.Contains(p.Name));

if (idProperty == null) return combo;

var listData = new List<int>();

foreach (var item in dbSet)

{

var idValue = idProperty.GetValue(item);

listData.Add(Convert.ToInt32(idValue));

}

combo.DataSource = listData;

if (currentValue != null)

{

combo.SelectedValue = currentValue;

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка загрузки данных: {ex.Message}");

Console.WriteLine($"Ошибка загрузки данных: {ex.Message}");

}

}

return combo;

}

private Control CreateSimpleInputControl(PropertyInfo prop,object propValue, int xPos, int yPos, bool readField)

{

int controlWidth = (int)(splitContainerEdit.Panel2.Width \* 0.4);

int labelWidth = splitContainerEdit.Panel2.Width – controlWidth – 30;

if (prop.PropertyType == typeof(bool) || prop.PropertyType == typeof(bool?))

{

return new CheckBox

{

Checked = (bool?)propValue ?? false,

Location = new Point(labelWidth + 20, yPos),

Width = controlWidth,

Height = 50,

Font = new Font("Segoe UI", 14),

TextAlign = ContentAlignment.MiddleLeft,

Enabled = readField

};

}

else

{

return new TextBox

{

Text = propValue?.ToString() ?? "",

Location = new Point(labelWidth + 20, yPos),

Width = controlWidth,

Height = 50,

Multiline = false,

Font = new Font("Segoe UI", 14),

ReadOnly = !readField

};

}

}

private void DisableIdFields()

{

foreach (Control control in splitContainerEdit.Panel2.Controls)

{

if (control is Label label && label.Text.StartsWith("ID"))

{

int index = splitContainerEdit.Panel2.Controls.IndexOf(label);

if (index < splitContainerEdit.Panel2.Controls.Count – 1)

{

var nextControl = splitContainerEdit.Panel2.Controls[index + 1];

if (nextControl is TextBoxBase)

{

nextControl.Enabled = false;

}

}

}

}

}

private bool IsCollectionProperty(PropertyInfo prop)

{

return prop.PropertyType.IsGenericType &&

(prop.PropertyType.GetGenericTypeDefinition() == typeof(ICollection<>) ||

prop.PropertyType.GetGenericTypeDefinition() == typeof(IEnumerable<>));

}

private void splitContainerEdit\_Panel2\_Resize(object sender, EventArgs e)

{

int controlHeight = 50;

foreach (Control control in splitContainerEdit.Panel2.Controls)

{

if (control is TextBox txt)

{

txt.Width = splitContainerEdit.Panel2.Width / 2 – 20;

var padding = (controlHeight – txt.PreferredHeight) / 2;

if (padding > 0)

{

txt.Margin = new Padding(0, padding, 0, 0);

}

}

else if (control is Label lbl)

{

lbl.Width = splitContainerEdit.Panel2.Width / 2 – 20;

}

}

}

private bool IsNavigationProperty(PropertyInfo prop)

{

if (prop.PropertyType.IsPrimitive || prop.PropertyType == typeof(string) || prop.PropertyType == typeof(DateTime))

return false;

return prop.PropertyType.Namespace == "WindowsFormsAppcCityBusStation.Models" ||

prop.PropertyType.Namespace?.StartsWith("System.Collections.Generic") == true;

}

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

{

if (buttonResultEdit.Text == "Сохранить изменения")

{

SaveChanges();

}

else if (buttonResultEdit.Text == "Удалить")

{

DeleteEntity();

}

else if (buttonResultEdit.Text == "Создать запись")

{

CreateEntity();

}

RefreshDataGridView();

}

private void SaveChanges()

{

using (var context = new CityBusStationEntities())

{

try

{

var entity = context.Set(currentEntityType).Find(currentEntityId);

if (entity == null)

{

MessageBox.Show("Запись не найдена в базе данных");

return;

}

int panelWidth = splitContainerEdit.Panel2.Width;

int controlWidth = panelWidth / 2;

foreach (Control control in splitContainerEdit.Panel2.Controls)

{

if (control is Label label)

{

try

{

var propName = label.Text.Trim(':');

var prop = currentEntityType.GetProperty(propName);

if (prop == null) continue;

var inputControl = splitContainerEdit.Panel2.Controls

.Cast<Control>()

.FirstOrDefault(c => c.Location.X > controlWidth &&

c.Location.Y == label.Location.Y);

if (inputControl == null) continue;

if (inputControl is TextBox textBox)

{

if (prop.PropertyType == typeof(string))

{

prop.SetValue(entity, textBox.Text);

}

else

{

try

{

var value = Convert.ChangeType(textBox.Text, prop.PropertyType);

prop.SetValue(entity, value);

}

catch

{

}

}

}

else if (inputControl is CheckBox checkBox)

{

prop.SetValue(entity, checkBox.Checked);

}

else if (inputControl is ComboBox comboBox && comboBox.SelectedValue != null)

{

var navProp = entity.GetType().GetProperty(propName);

if (navProp != null)

{

var navEntityType = navProp.PropertyType;

var setMethod = typeof(DbContext).GetMethod("Set", Type.EmptyTypes)

.MakeGenericMethod(navEntityType);

dynamic dbSet = setMethod.Invoke(context, null);

var navEntity = dbSet.Find(comboBox.SelectedValue);

if (navEntity != null)

{

navProp.SetValue(entity, navEntity);

}

}

}

}

catch (Exception ex)

{

Debug.WriteLine($"Ошибка при сохранении свойства {label.Text}: {ex.Message}");

}

}

}

context.SaveChanges();

MessageBox.Show("Изменения сохранены успешно!");

RefreshDataGridView();

}

catch (DbUpdateException dbEx)

{

MessageBox.Show($"Ошибка базы данных при сохранении: {dbEx.InnerException?.Message ?? dbEx.Message}");

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при сохранении: {ex.Message}");

}

}

}

private void DeleteEntity()

{

using (var context = new CityBusStationEntities())

{

try

{

var entity = context.Set(currentEntityType).Find(currentEntityId);

if (entity != null)

{

context.Set(currentEntityType).Remove(entity);

context.SaveChanges();

MessageBox.Show("Запись удалена успешно!");

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при удалении: {ex.Message}");

}

}

}

private void CreateEntity()

{

using (var context = new CityBusStationEntities())

{

try

{

dynamic newEntity = Activator.CreateInstance(currentEntityType);

context.Set(currentEntityType).Add(newEntity);

FillEntityFromControlsForCreate(context, newEntity);

context.SaveChanges();

MessageBox.Show("Запись успешно создана!");

RefreshDataGridView();

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при создании: {ex.Message}");

Console.WriteLine(ex.Message);

}

}

}

private void AddDynamicFormForCreate(dynamic entity)

{

splitContainerEdit.Panel2.Controls.Clear();

int yPos = 10;

int panelWidth = splitContainerEdit.Panel2.Width;

int controlWidth = panelWidth / 2;

using (var context = new CityBusStationEntities())

{

foreach (var prop in entity.GetType().GetProperties())

{

if (prop.Name.StartsWith("ID") ||

IsCollectionProperty(prop) ||

IsNavigationProperty(prop))

continue;

var lbl = new Label

{

Text = $"{prop.Name}:",

Location = new Point(10, yPos),

Width = controlWidth – 20,

Height = 30,

Font = new Font("Segoe UI", 14)

};

splitContainerEdit.Panel2.Controls.Add(lbl);

object propValue = prop.GetValue(entity);

Control inputControl;

string navigationPropertyName;

if (propValue == null)

{

if (prop.PropertyType == typeof(string))

propValue = string.Empty;

else if (prop.PropertyType == typeof(int))

propValue = 0;

else if (prop.PropertyType == typeof(bool))

propValue = false;

else if (prop.PropertyType == typeof(DateTime))

propValue = DateTime.Now;

}

if (\_navigationMappings.TryGetValue(prop.Name, out navigationPropertyName))

{

inputControl = CreateNavigationComboBox(

context,

navigationPropertyName,

propValue,

controlWidth,

yPos,

true);

}

else

{

inputControl = CreateSimpleInputControl(

prop,

propValue,

controlWidth,

yPos,

true);

}

if (inputControl != null)

{

splitContainerEdit.Panel2.Controls.Add(inputControl);

yPos += 35;

}

}

}

}

private void FillEntityFromControlsForCreate(CityBusStationEntities context, dynamic entity)

{

int panelWidth = splitContainerEdit.Panel2.Width;

int controlWidth = panelWidth / 2;

foreach (Control control in splitContainerEdit.Panel2.Controls)

{

if (control is Label label)

{

var propName = label.Text.Trim(':');

var prop = currentEntityType.GetProperty(propName);

if (prop == null || propName.StartsWith("ID")) continue;

var inputControl = splitContainerEdit.Panel2.Controls

.Cast<Control>()

.FirstOrDefault(c => c.Location.X > controlWidth &&

c.Location.Y == label.Location.Y);

if (inputControl == null) continue;

if (inputControl is ComboBox comboBox && comboBox.SelectedValue != null)

{

prop.SetValue(entity, comboBox.SelectedValue);

}

if (inputControl is TextBox textBox)

{

SetPropertyFromString(entity, prop, textBox.Text);

}

else if (inputControl is CheckBox checkBox)

{

prop.SetValue(entity, checkBox.Checked);

}

else if (inputControl is NumericUpDown numericUpDown)

{

prop.SetValue(entity, (int)numericUpDown.Value);

}

else if (inputControl is DateTimePicker dateTimePicker)

{

prop.SetValue(entity, dateTimePicker.Value);

}

}

}

}

private void SetPropertyFromString(dynamic entity, PropertyInfo prop, string value)

{

if (string.IsNullOrEmpty(value) && prop.PropertyType != typeof(string))

{

return;

}

try

{

if (prop.PropertyType == typeof(string))

{

prop.SetValue(entity, value);

}

else if (prop.PropertyType == typeof(int))

{

if (int.TryParse(value, out int intValue))

prop.SetValue(entity, intValue);

}

else if (prop.PropertyType == typeof(DateTime))

{

if (DateTime.TryParse(value, out DateTime dateValue))

prop.SetValue(entity, dateValue);

}

else if (prop.PropertyType == typeof(bool))

{

if (bool.TryParse(value, out bool boolValue))

prop.SetValue(entity, boolValue);

}

}

catch (Exception ex)

{

Debug.WriteLine($"Ошибка преобразования значения для {prop.Name}: {ex.Message}");

}

}

private void SetNavigationProperty(CityBusStationEntities context, dynamic entity, PropertyInfo prop, object selectedValue)

{

var navProp = entity.GetType().GetProperty(prop.Name);

if (navProp != null)

{

var navEntityType = navProp.PropertyType;

var setMethod = typeof(DbContext).GetMethod("Set", Type.EmptyTypes)

.MakeGenericMethod(navEntityType);

dynamic dbSet = setMethod.Invoke(context, null);

var navEntity = dbSet.Find(selectedValue);

if (navEntity != null)

{

navProp.SetValue(entity, navEntity);

}

}

}

private void RefreshDataGridView()

{

if (currentEntityType.Name.Contains("Tickets"))

{

LoadTicketsData();

}

else if (currentEntityType.Name.Contains("Schedules"))

{

LoadSchedulesData();

}

else if (currentEntityType.Name.Contains("Drivers"))

{

LoadDriversData();

}

else if (currentEntityType.Name.Contains("Buses"))

{

LoadBusesData();

}

else if (currentEntityType.Name.Contains("Routes"))

{

LoadRoutesData();

}

else if (currentEntityType.Name.Contains("Destinations"))

{

LoadPointsData();

}

else

{

MessageBox.Show("Неизвестный тип данных для обновления");

ShowPanelMain\_Click(null, null);

}

}

private object GetCurrentEntityFromControls(int panelWidth)

{

var controlWidth = panelWidth / 2;

var entity = Activator.CreateInstance(currentEntityType);

foreach (Control control in splitContainerEdit.Panel2.Controls)

{

if (control is Label label)

{

var propName = label.Text.Trim(':');

var prop = currentEntityType.GetProperty(propName);

if (prop == null) continue;

var inputControl = splitContainerEdit.Panel2.Controls

.Cast<Control>()

.FirstOrDefault(c => c.Location.X > controlWidth &&

c.Location.Y == label.Location.Y);

if (inputControl != null)

{

if (inputControl is TextBox textBox)

{

SetPropertyValue(entity, prop, textBox.Text);

}

else if (inputControl is CheckBox checkBox)

{

prop.SetValue(entity, checkBox.Checked);

}

else if (inputControl is ComboBox comboBox)

{

prop.SetValue(entity, comboBox.SelectedValue);

}

}

}

}

return entity;

}

private void SetPropertyValue(object entity, PropertyInfo prop, string value)

{

if (prop.PropertyType == typeof(string))

{

prop.SetValue(entity, value);

}

else if (prop.PropertyType == typeof(int))

{

if (int.TryParse(value, out int intValue))

prop.SetValue(entity, intValue);

}

}

}

}

Приложение В

(справочное)

**Код Unit-тестов**

using Microsoft.VisualStudio.TestTools.UnitTesting;

using System;

using WindowsFormsAppcCityBusStation;

using System.IO;

using WindowsFormsAppcCityBusStation.Models;

namespace UnitTestProject1

{

[TestClass]

public class UnitTest1

{

private Form1 \_form = new Form1();

[TestMethod]

public void TestExportEntityDataToFile()

{

Type entityType = typeof(WindowsFormsAppcCityBusStation.Models.Schedules);

string testFilePath = Path.Combine(Path.GetTempPath(), "test\_export.txt");

string expected = "ОК";

string result = \_form.ExportEntityDataToFile(entityType, testFilePath);

Assert.AreEqual(expected, result);

}

[TestMethod]

public void TestEntityDataToFile()

{

string testDir = @"C:\Office\";

string testFilePath = Path.Combine(testDir, "test\_export.txt");

Assert.IsTrue(File.Exists(testFilePath));

}

[TestMethod]

public void TestGetEntityByIdIsNotNull()

{

Type entityType = typeof(Buses);

int testId = 1;

var privateObject = new PrivateObject(\_form);

var result = privateObject.Invoke("GetEntityById", entityType, testId);

Assert.IsNotNull(result);

}

[TestMethod]

public void TestGetEntityByIdType()

{

Type entityType = typeof(Buses);

int testId = 1;

var privateObject = new PrivateObject(\_form);

var result = privateObject.Invoke("GetEntityById", entityType, testId);

Assert.IsInstanceOfType(result, typeof(Buses));

}

[TestMethod]

public void TestCreateEntity()

{

Buses expectedBus = new Buses

{

IDBus = 1,

BusNumber = "TEST-123",

NumberOfSeats = 50,

Carrier = 1,

FireExtinguisher = true

};

var privateObject = new PrivateObject(\_form);

var result = privateObject.Invoke("CreateEntity") as Buses;

Assert.IsNotNull(result);

Assert.AreEqual(expectedBus.IDBus, result.IDBus);

Assert.AreEqual(expectedBus.BusNumber, result.BusNumber);

Assert.AreEqual(expectedBus.NumberOfSeats, result.NumberOfSeats);

Assert.AreEqual(expectedBus.Carrier, result.Carrier);

Assert.AreEqual(expectedBus.FireExtinguisher, result.FireExtinguisher);

}

}

}

Приложение Г

(справочное)

**Тест-кейсы**

Таблица Г.1 – Тестовый пример 1

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Успешная аутентификация пользователя | | |
| **Предусловие: Приложение запущено. Открыто окно авторизации** | | | |
| Удостовериться, что в базе данных есть пользователь с именем пользователя «qwert» и паролем «12345» | | | |
| **Действие** | | **Ожидаемый результат** | **Статус** |
| Ввести имя пользователя, пароль:   * Логин: «qwert» * Пароль: «12345» | | На экране в полях отображаются введенные значения:   * Логин: «qwert» * Пароль: «12345» | Пройден |
| Нажать кнопку «Войти». | | Форма скрывается, и появляется окно с функционалом администратора, что подтверждает успешное авторизацию | Пройден |

Таблица Г.2 – Тестовый пример 2

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Аутентификация с отсутствием вводимых данных | | |
| **Предусловие: Приложение запущено. Открыто окно авторизации** | | | |
| **Действие** | | **Ожидаемый результат** | **Статус** |
| Не вводя в поля никаких данных нажать на кнопку «Войти» | | Выведется сообщение «Пользователь с таким логином не найдет» | Пройден |

Таблица Г.3 – Тестовый пример 3

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Добавление нового водителя | | |
| **Предусловие: Приложение запущено. Открыто окно создания записи о новом водителе** | | | |
| Удостовериться, что форма создания нового водителя отображается. | | | |
| **Действие** | | **Ожидаемый результат** | **Cтатус** |
| Ввести в следующие поля следующие значения:  Surname: «Подосенов»,  Name: «Тихон»,  Patronimyc: «Романович»,  Age: 32,  YearsOfDrivingExperience: 10 | | Поля заполняются следующими данными:  Surname: «Подосенов»,  Name: «Тихон»,  Patronimyc: «Романович»,  Age: 32,  YearsOfDrivingExperience: 10 | Пройден |
| Нажать кнопку «Создать». | | Выводится сообщение «Запись успешно создана!» | Пройден |

Таблица Г.4 – Тестовый пример 4

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Удаление записи о билете | | |
| **Предусловие: Приложение запущено. Открыто окно удаления записи о билете с ID 3** | | | |
| Удостовериться, что форма удаление записи о билете с ID 3 отображается. | | | |
| **Действие** | | **Ожидаемый результат** | **Cтатус** |
| Нажать на кнопку «Удалить» | | Выведется сообщение «Запись удалена успешно!». Откроется раздел «Билеты» | Пройден |

Таблица Г.5 – Тестовый пример 5

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Выход из учетной записи системного администратора | | |
| **Предусловие: Приложение запущено. Произведен вход под учетной записью системного администратора с логином «qwert» и паролем «12345»** | | | |
| Удостовериться, что вход произведен через учетную запись системного администратора. | | | |
| **Действие** | | **Ожидаемый результат** | **Cтатус** |
| Перейти на главный раздел и нажать на кнопку «Выйти» | | Произойдет выход из учетной записи системного администратора и откроется раздел с авторизацией | Пройден |