Министерство образования и науки Республики Башкортостан

Государственное автономное профессиональное образовательное учреждение

Уфимский колледж статистики, информатики и вычислительной техники

|  |  |
| --- | --- |
|  | УТВЕРЖДАЮ  Заместитель директора  по учебной работе  \_\_\_\_\_\_\_\_\_\_\_З.З. Курмашева  « » 2024 г. |

ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ И РАЗРАБОТКА ПРИЛОЖЕНИЯ

ДЛЯ ОФОРМЛЕНИЯ ПРОДАЖИ В АВТОСАЛОНЕ

Пояснительная записка к курсовому проекту

МДК 11.01 Технология разработки и защиты баз данных

|  |  |  |
| --- | --- | --- |
|  | Руководитель проекта  Р.Ф. Каримова  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г. | Руководитель проекта  И.И. Иванов  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г. |
|  | Студент гр. 21П-1  М.А. Камалов  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г. | Студент гр. 21П-1  Е.А. Одинцов  » \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г. |

2024

Министерство образования и науки Республики Башкортостан

Государственное автономное профессиональное образовательное учреждение

Уфимский колледж статистики, информатики и вычислительной техники

|  |  |
| --- | --- |
|  | УТВЕРЖДАЮ  Заместитель директора  по учебной работе  З.З. Курмашева  « » 2024 г. |

ЗАДАНИЕ

на курсовой проект студенту дневного отделения, группы 21П-1, специальности 09.02.07 Информационные системы и программирование

Фамилия, имя, отчество: Камалов Максим Альфредович

Тема курсового проекта: «Проектирование базы данных для оформления продажи в автосалоне».

Текст задания:

при выполнении курсового проекта должны быть решены следующие задачи:

1. спроектирована структура базы данных;
2. разработана структура программы;
3. реализованы возможность получение списков автомобилей по различным параметрам, просмотр полной информации об автомобилях и покупателей, возможность составлять договор купли-продажи.

В результате выполнения курсового проекта должны быть представлены:

1. пояснительная записка, состоящая из следующих разделов:

Введение

1 Постановка задачи

2 Экспериментальный раздел

Заключение

Приложения

Список сокращений

Список использованных источников

1. электронный носитель, содержащий разработанный программный продукт;
2. презентация курсового проекта в электронном виде.

Список рекомендуемых источников:

1. Learn.Microsoft [Электронный ресурс] / Туториалы по WPF и C#. – Режим доступа https://learn.microsoft.com, свободный
2. Stackoverflow [Электронный ресурс] / Ответы на вопросы. – Режим доступа https://stackoverflow.com/, свободный
3. Github [Электронный ресурс] / Примеры кода. – Режим доступа https://github.com/, свободный

Задание к выполнению получил «31» января 2024 г.

Студент Камалов Максим Альфредович

Срок окончания «31» мая 2024 г.

Руководитель курсового проекта Р.Ф. Каримова

Задание рассмотрено на заседании цикловой комиссии информатики

«11» января 2024 г.

Председатель цикловой комиссии информатики О.В.Фатхулова

Министерство образования и науки Республики Башкортостан

Государственное автономное профессиональное образовательное учреждение

Уфимский колледж статистики, информатики и вычислительной техники

ЗАКЛЮЧЕНИЕ

на курсовой проект

Студент Камалов Максим Альфредович

Группа 21П-1

Специальность 09.02.07 Информационные системы и программирование

Тема Проектирование базы данных для оформления продажи в автосалоне

Объем курсового проекта:

количество листов пояснительной записки

количество листов графической части

Заключение о степени соответствия заданию на курсовое проектирование

Характеристика качеств, проявленных студентом при работе над проектом: самостоятельность, дисциплинированность, умение планировать работу и пользоваться литературным материалом и т.д.

Положительные стороны курсового проекта

Недостатки курсового проекта

Характеристика общетехнической и специальной подготовки студента

Заключение и предлагаемая оценка за курсовой проект

Руководитель курсового проекта Каримова Резида Флюновна

«\_\_\_\_» 2024 г.

Подпись

АННОТАЦИЯ

Пояснительная записка к курсовому проекту содержит постановку и программу решения задачи «Проектирование базы данных и разработка программы для оформления продажи в автосалоне».

Программа AutoMir написана на языке C# с использованием системы управления базами данных MySQL, а также с использованием фреймворка WPF, предназначена для работы в операционной системе Windows 10 и выше, отлажена на данных контрольного примера.

СОДЕРЖАНИЕ

|  |  |
| --- | --- |
|  | лист |
| Введение | 4 |
| 1 Постановка задачи | 5 |
| 1.1 Описание предметной области | 5 |
| 1.2 Описание входной информации | 8 |
| 1.3 Описание выходной информации | 8 |
| 1.4 Концептуальное моделирование | 16 |
| 1.5 Логическое моделирование | 16 |
| 1.6 Описание структуры базы данных | 18 |
| 1.7 Контрольный пример | 20 |
| 1.8 Общие требования к программному продукту | 20 |
| 2 Экспериментальный раздел | 22 |
| [2.1 Описание программы](file:///C:\Users\Intel\Downloads\Документация_1часть_АвхадиевКамиль%20(1).docx#Описаниепрограммы) | 26 |
| 2.2 Протокол тестирования программного продукта | 25 |
| 2.3 Руководство пользователя | 28 |
| Заключение | 35 |
| Приложения А. ппппп | 36 |
| Список использованных источников | 58 |

ВВЕДЕНИЕ

В современном мире базы данных играют ключевую роль в организации и управлении информацией, особенно в сфере коммерции и торговли. Автомобильная индустрия не является исключением, где эффективное управление данными о продажах автомобилей становится критически важным для успешного функционирования предприятий. Для автодилеров и автопроизводителей наличие надежной и эффективной базы данных о продажах автомобилей является необходимостью для оптимизации процессов продаж, управления инвентарем и удовлетворения потребностей клиентов.

Цель данной курсовой работы состоит в исследовании и анализе методов и технологий управления базой данных для продажи автомобилей. В рамках данного исследования будут рассмотрены основные аспекты проектирования, реализации и оптимизации баз данных, специфичные для автомобильной индустрии. Кроме того, будет проведен анализ существующих решений и применяемых практик в данной области с целью выявления наиболее эффективных подходов и инструментов.

Итак, данная курсовая работа представляет собой попытку систематизировать и проанализировать основные аспекты управления базой данных в контексте продажи автомобилей, а также выделить лучшие практики и технологии для достижения оптимальных результатов в данной области.

Задачами курсового проекта являются:

* описать предметную область;
* разработать структуру базы данных;
* разработать приложение;
* провести тестирование приложения.

1. Постановка задачи

1.1 Описание предметной области

Предметная область «Продажа автомобилей» охватывает широкий спектр деятельности, связанный с торговлей новыми и подержанными автомобилями. В данной предметной области важно учитывать не только информацию о самом автомобиле, но и данные о покупателях, их предпочтениях, контактных данных, а также взаимодействие с поставщиками и перегонщиками автомобилей. Вот некоторые ключевые аспекты предметной области "Продажа автомобилей":

1. Автоматизированный подбор вариантов для покупателя:

* система фильтрации по параметрам автомобилей, соответствующих требованиям покупателя.
* генерация списка подходящих вариантов для просмотра покупателем.

2. Заявки для поставщиков и перегонщиков автомобилей:

- Формирование заявок на приобретение автомобилей у поставщиков.

- Формирование заявок на перегон автомобилей из других регионов или стран.

3. Меню приложения и средства диалога:

- Меню с разделением на функциональные блоки («Каталог Авто», «Покупатель», «Заявки»).

- Интерфейс для взаимодействия с базой данных, включая формы ввода и изменения данных.

4. Запросы и договара:

- Запросы для извлечения конкретной информации из базы данных (например, список автомобилей определенной марки с заданным диапазоном цен).

- Формирования договара купли-продажи для вывода на печать ключевой информации.

В базе данных должны храниться следующие сущности: автомобили, покупатели, продажи.

В таблице «Автомобили» содержится следующая информация:

- ID авто.

- ID бренда.

- Модель автомобиля.

- Страна производства.

- VIN-номер.

- Номер ПТС.

- Год выпуска.

- Мощность двигателя.

- Тип трансмиссии.

- Комплектация.

- Цвет.

- ID кузова.

- ID состояния

- ID статус (в продаже, продан).

- Пробег.

- Количество владельцев.

- Фото.

- Запрашиваемая цена.

В таблице «Покупатели» содержится следующая информация:

- ID покупателя.

- ФИО.

- Номер телефона.

- Адрес эл. почты

- Адрес проживания.

- Серия паспорта.

- Номер паспорта

- Требования к марке автомобиля.

- Требования к техническим характеристикам и состоянию автомобиля.

- Допустимая цена автомобиля (Бюджет клиента).

Продажи. После каждой сделки с покупателем нужно иметь сведения об покупки авто покупателем именно в этом автосалоне. Нужно это для дальнейшего технического обслуживания ТС через определенный промежуток времени, а также при поломке определенного узла автомобиля иметь возможность гарантийного сервиса для владельца авто.

В таблице «Продажи» содержится следующая информация:

- ID продажи.

- ID авто.

- ID покупателя.

- Цена авто (фиксированная цена авто при продаже).

- Гарантия (кол-во месяцев).

- Дата продажи (сделки с покупателем).

Предметная область "Продажа автомобилей" требует разработки системы управления данными, которая обеспечит эффективное ведение информации о автомобилях, покупателях и операциях с ними, с целью улучшения процессов продаж и удовлетворения потребностей клиентов.

Входными данными являются данные о автомобиле и покупателя. Выходными данными являются договор купли-продажи авто, договор о передачи авто новому владельцу.

1.2 Описание входной информации

Входной информацией для выполнения задачи являются справочники: Автомобили, Покупатели, Продажи.

1.3 Описание выходной информации

Выходной информацией является различные договоры:

- Договор купли-продажи.

- Информация об автомобиле.

- Список автомобилей.

- Информация о покупателе.

- Список покупателей.

- Список продаж

Описание выходных документов представлено в таблице 1.3.1.

Таблица 1.3.1 – Описание выходных документов

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Наименование документа (шифр) | Периодичность выдачи документа | Кол-во экз. | Куда передаются | Поля сортировки | Поля группировки | Итоги |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Информация об автомобиле | По мере необходимости | 1 | Сотруднику | - | - | - |
| Список автомобилей | По мере необходимости | 1 | Клиенту, Сотруднику | Кузов, Бренд, Состояние,КПП | - | - |
| Информация о покупателе | По мере необходимости | 1 | Сотруднику | - | - | - |
| Список покупателей | По мере необходимости | 1 | Сотруднику | - | - | - |
| Список продаж | По мере необходимости | 1 | Сотруднику | По дате создания | - | - |

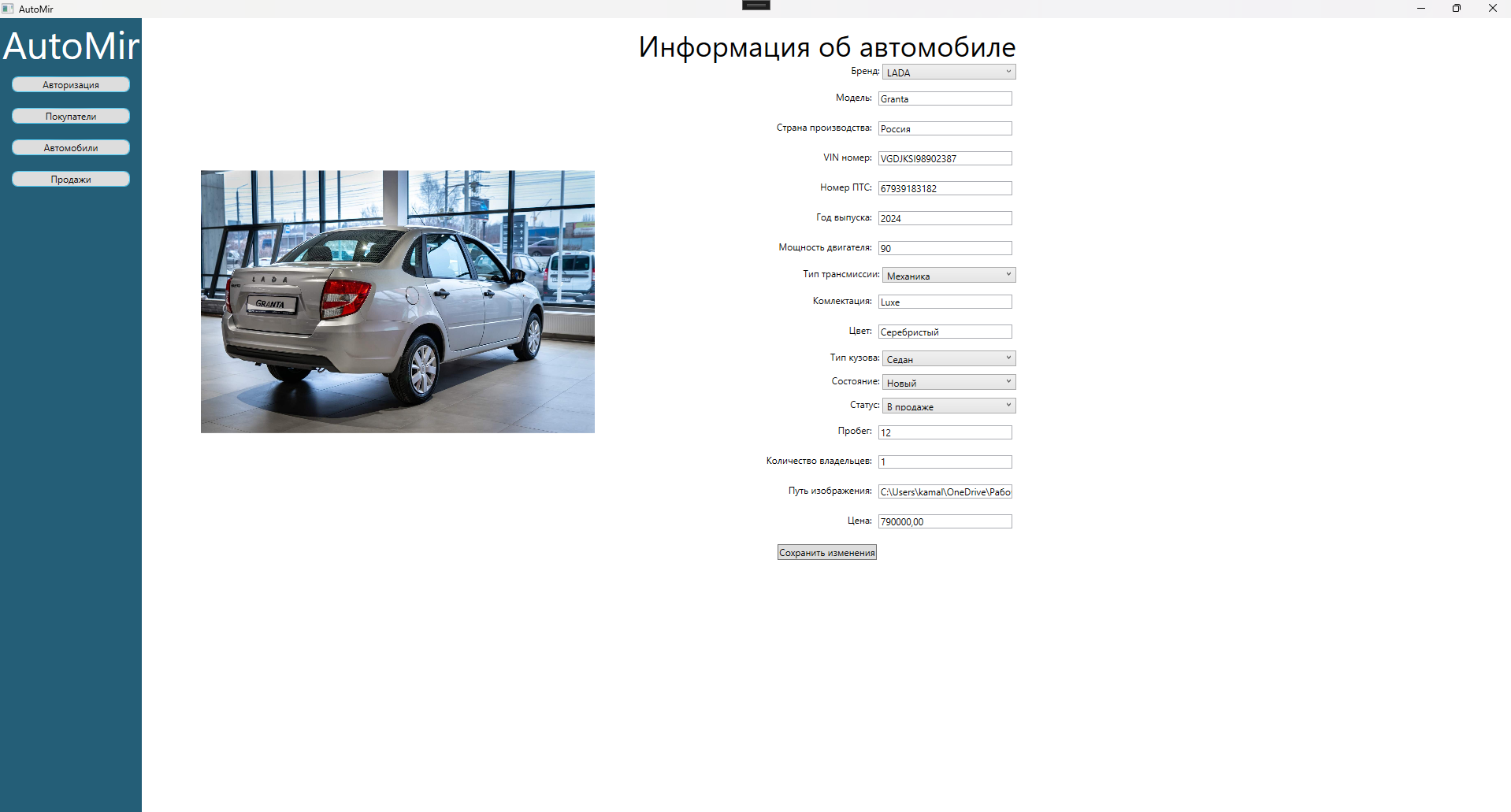
Рисунок 1.3.2 – Пример выходного документа «Информация об автомобиле» 

Рисунок 1.3.3 – Пример выходного документа «Список автомобилей»

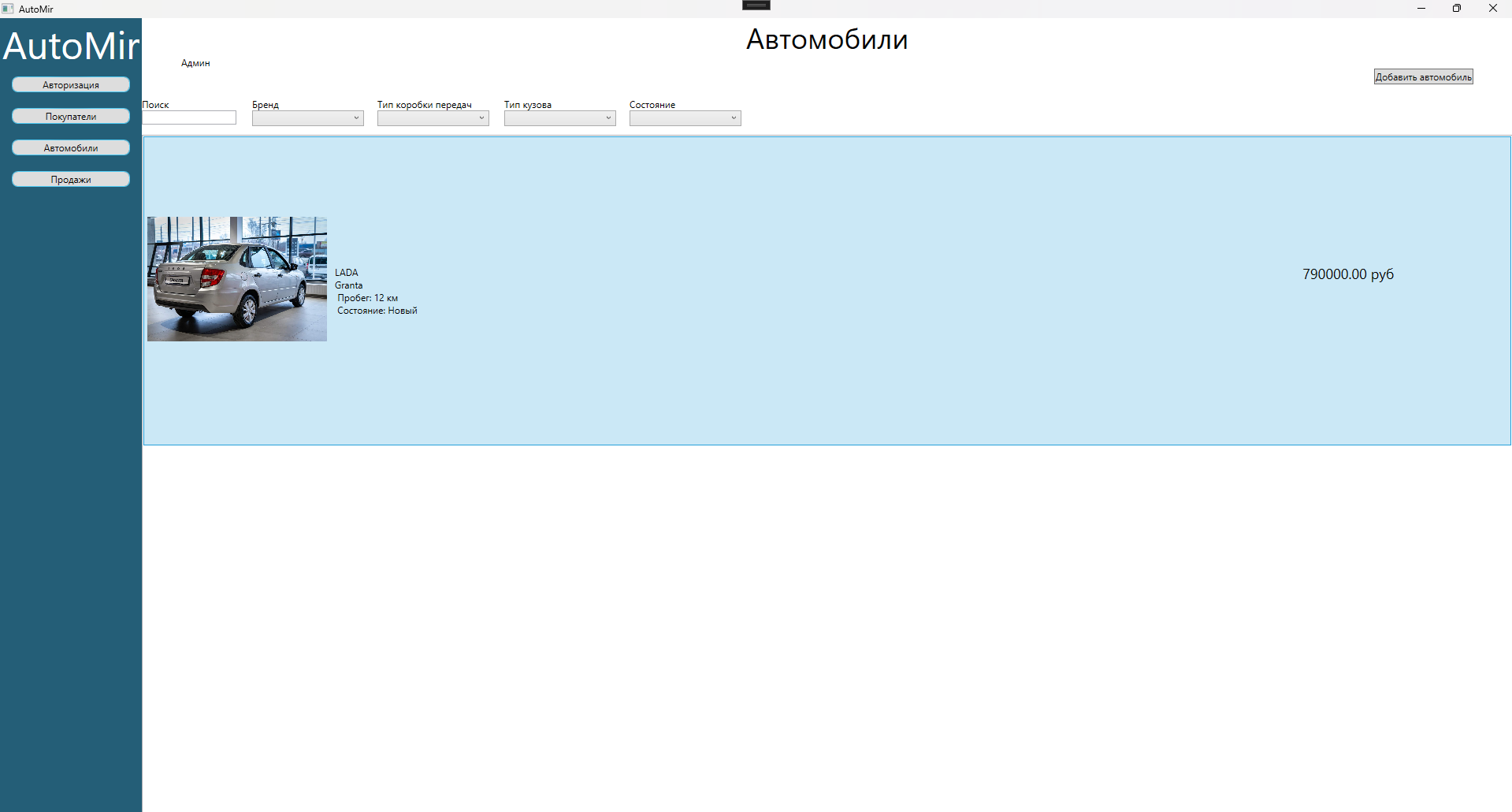


Рисунок 1.3.4 – Пример выходного документа «Информация о покупателе»

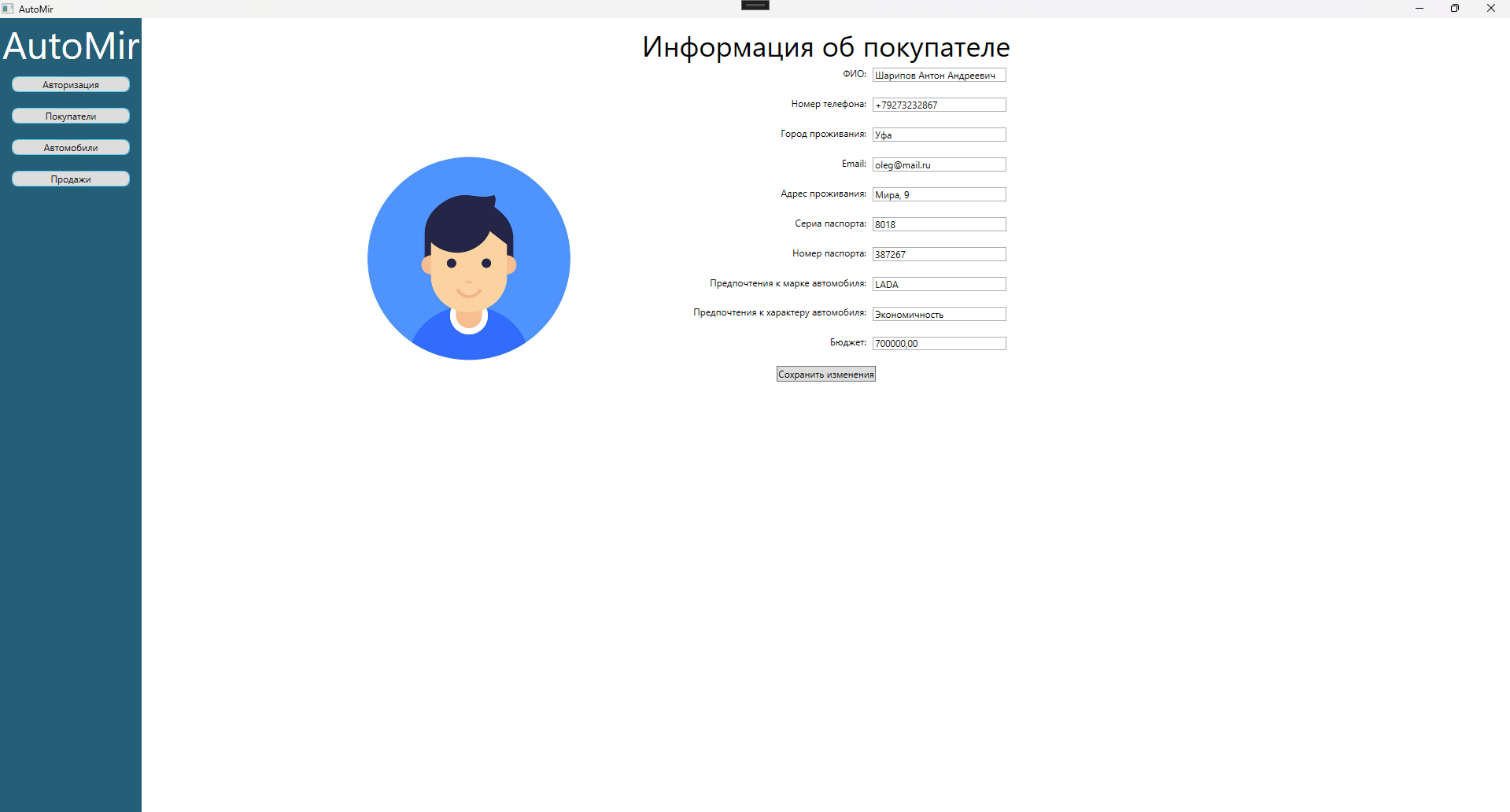


Рисунок 1.3.5 – Пример выходного документа «Список покупателей»

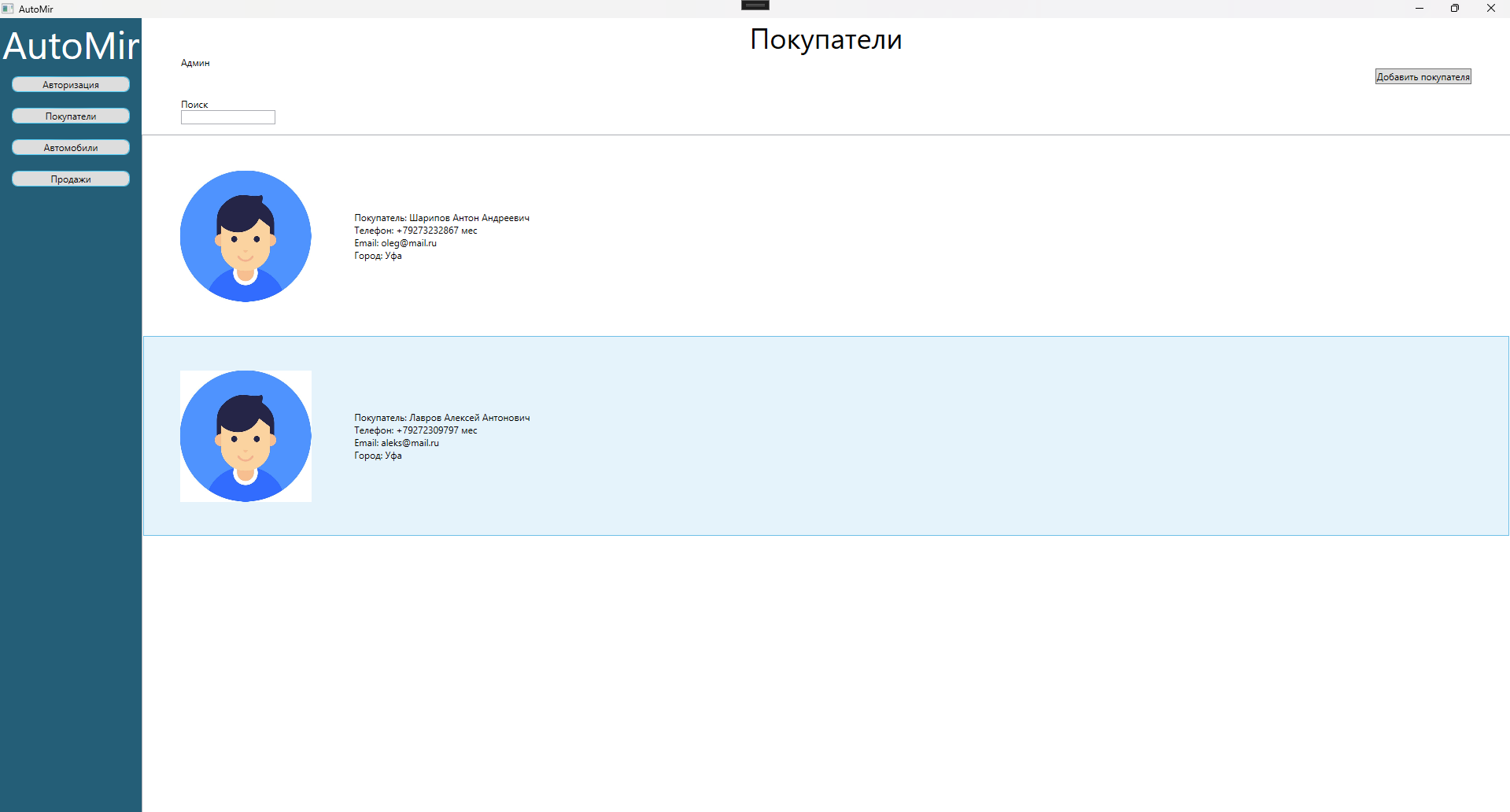


Рисунок 1.3.6 – Пример выходного документа «Список продаж»

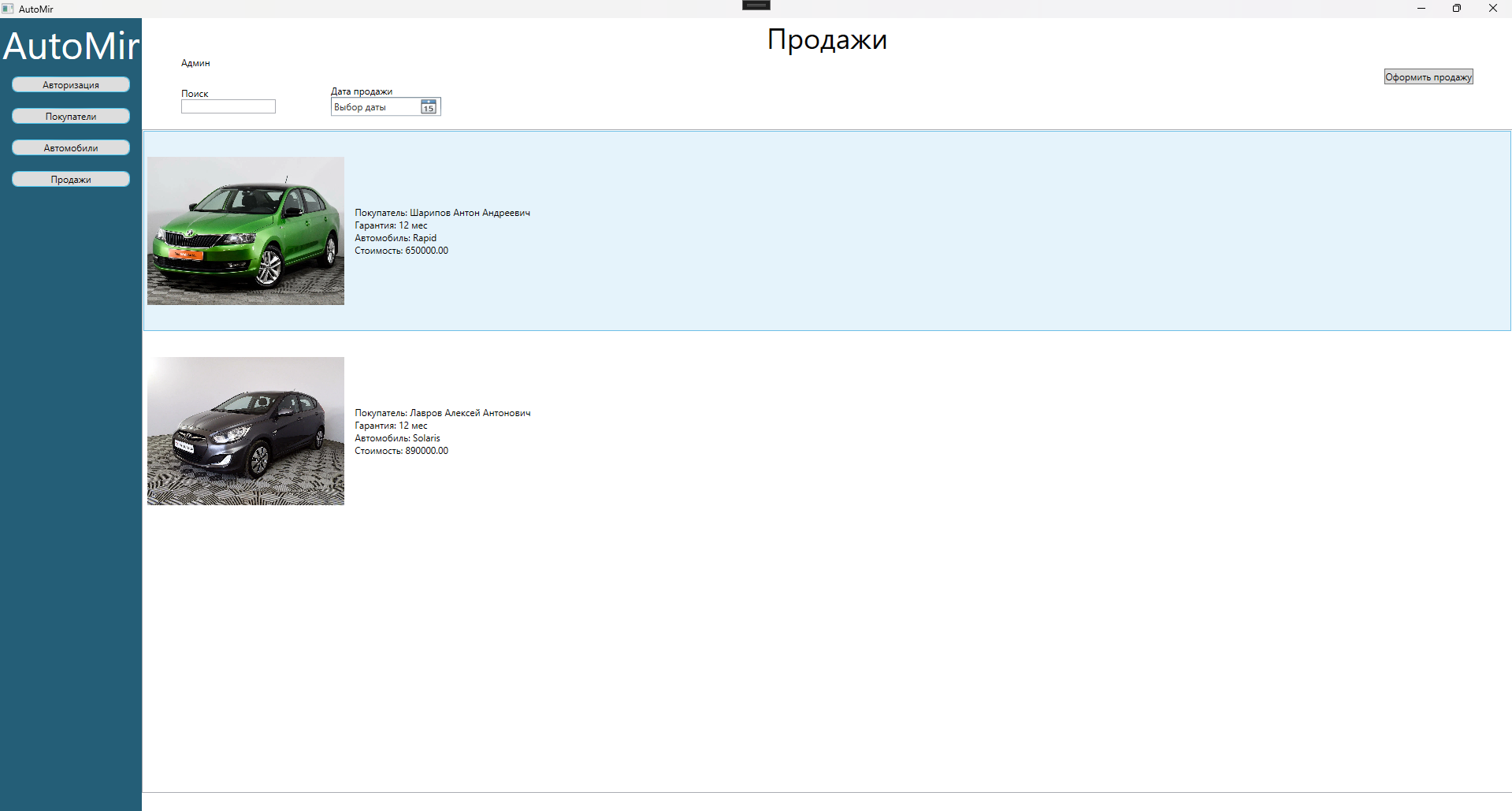
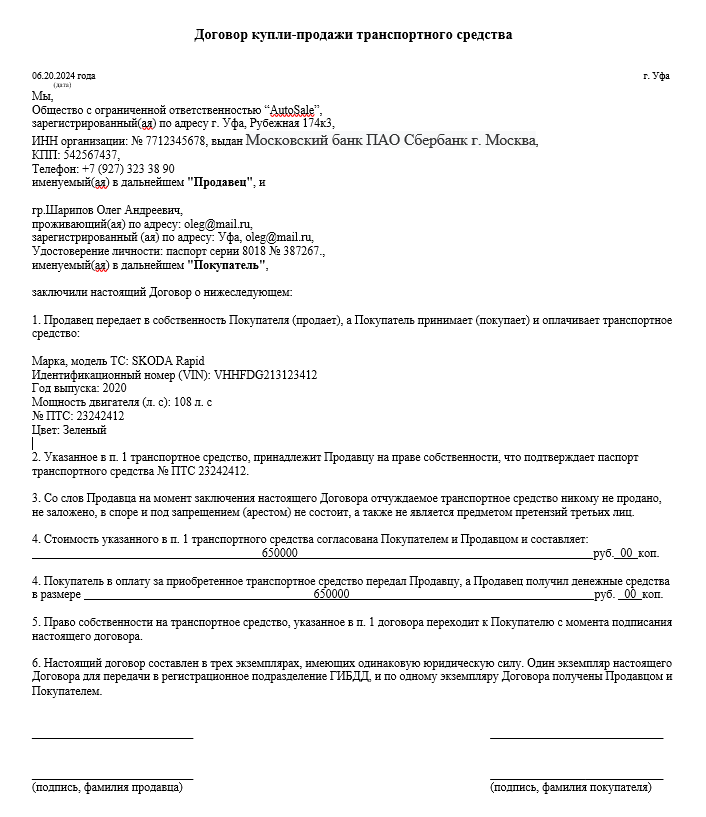


Рисунок 1.3.7 – Шаблон выходного документа «Договор купли-продажи»



1.4 Концептуальное проектирование

Концептуальная модель базы данных - это некая наглядная диаграмма, нарисованная в принятых обозначениях и подробно показывающая связь между объектами и их характеристиками. Создается концептуальная модель для дальнейшего проектирования базы данных и перевод ее, например, в реляционную базу данных. На концептуальной модели в визуально удобном виде прописываются связи между объектами данных и их характеристиками. В концептуальной модели есть принятые обозначения элементов. Сущность или объект обозначать прямоугольником, отношения обозначать ромбом, атрибуты объектов, обозначаются овалом. Если сущность связана с отношением, то их связь обозначается прямой линией со стрелкой. Необязательная связь обозначается пунктирной линией. Мощная связь обозначается двойной линией.

Концептуальная модель базы данных представлена в рисунке 1.4.1

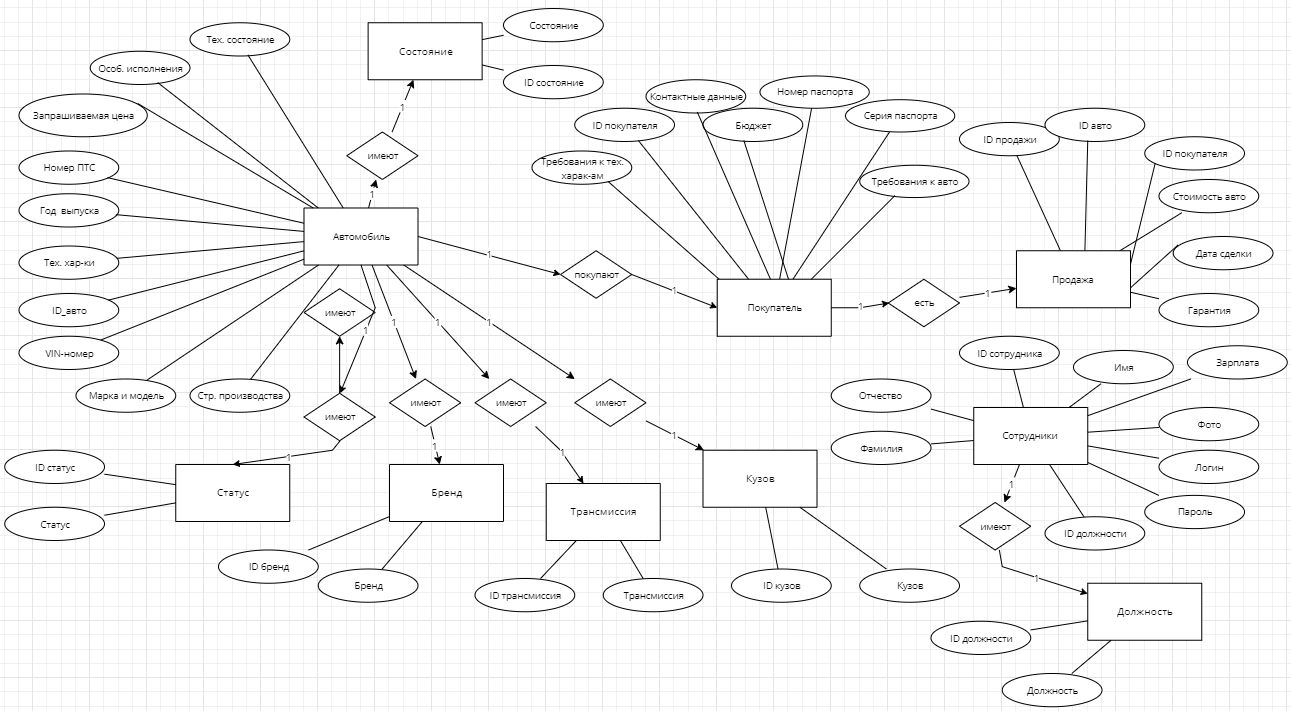


Рисунок 1.4.1 – Концептуальная модель базы данных

1.5. Логическое моделирование

При логическом моделировании происходит окончательное определение структуры данных, определяются ограничения, накладываемые на эти данные, целью которых является обеспечить целостность данных. Наиболее распространенной моделью данных является реляционная модель. В этой модели данных каждая сущность представляется в виде таблицы.

Логическое моделирование заключается в переходе от концептуальной модели к взаимосвязанным таблицам. Этот переход состоит из следующих шагов:

1. Преобразование сущностей:

- каждая простая сущность становится таблицей.

- каждый атрибут становится столбцом таблицы.

- уникальный идентификатор сущности становится ключом таблицы.

2. Преобразование связи:

- сущности, связанные обязательной связью один к одному можно объединить в одну таблицу.

- связи типа один к одному возможные и связи типа один ко многим реализуются путем переноса ключевых атрибутов таблиц, соответствующих сущностей, стоящих со стороны один в таблице соответствующих сущностей, стоящих со стороны многие.

- связи типа многие ко многим реализуются при помощи промежуточных таблиц, содержащих ключевые атрибуты связываемых таблиц в качестве внешних ключей.

Схема данных – это структура базы данных, описанная на формальном языке, поддерживаемом СУБД (системой управления базы данных). В реляционных базах данных схема определяет таблицы, поля в каждой таблице и ограничения целостности, такие как первичный и внешний ключи.

Схема данных представлена на рисунке 1.5.1.

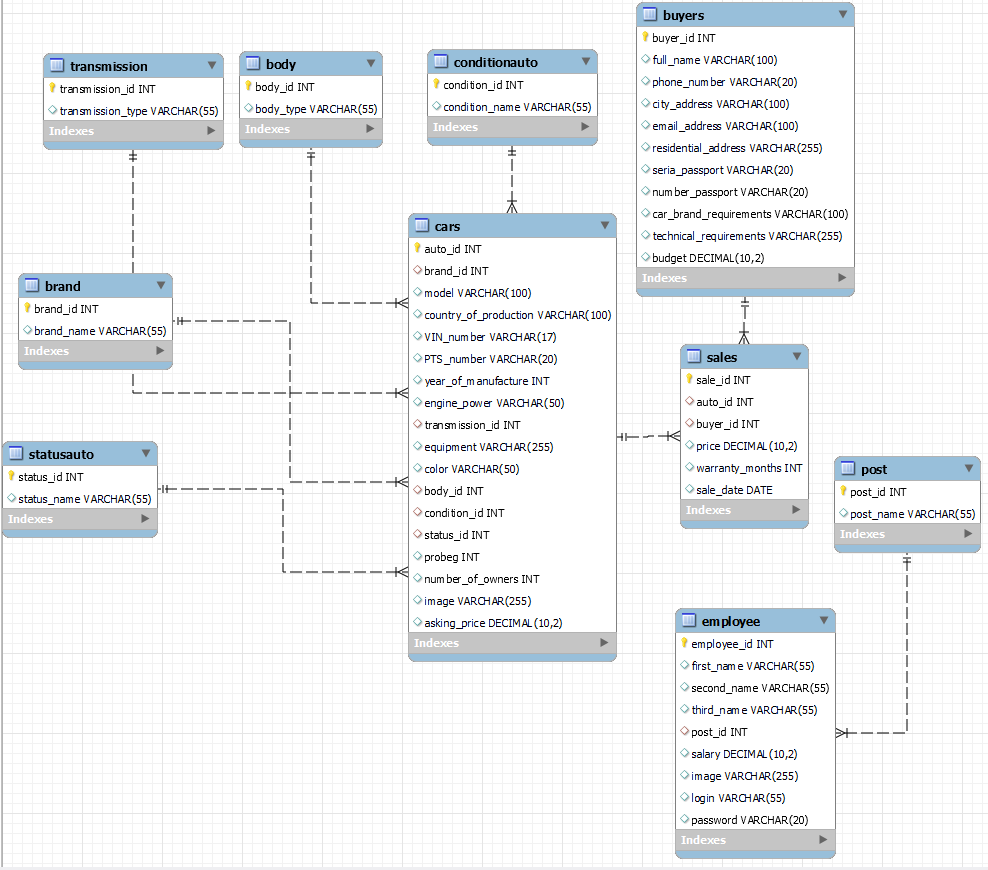


Рисунок 1.5.1 – Схема данных БД

1.6 Описание структуры базы данных

Описание структуры базы данных представлено в таблицах 1.6.1 – 1.6.10.

Таблица 1.6.1 - Car (Автомобили)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Описание поля | Тип данных | Размер поля | Тип ключа |
| auto\_id | ID авто | INT |  | PK |
| brand\_id | Марка и модель | VARCHAR | 100 | FK |
| model | Модель | VARCHAR | 100 |  |
| country\_of\_production | Страна производства | VARCHAR | 100 |  |

Продолжение таблицы 1.6.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| VIN\_number | Вин-номер | VARCHAR | 17 |  |
| PTS\_number | Номер ПТС | VARCHAR | 20 |  |
| year\_of\_manufacture | Год выпуска | INT |  |  |
| engine\_power | Мощность двигателя | VARCHAR | 50 |  |
| transmission\_id | Тип кпп | INT |  | FK |
| equipment | Комлектация | VARCHAR | 255 |  |
| color | Цвет | VARCHAR | 255 |  |
| body\_id | Тип кузова | INT |  | FK |
| condition\_id | Тех. состояние | INT |  | FK |
| status\_id | Статус | INT |  | FK |
| probeg | Пробег | INT |  |  |
| number\_of\_owners | Кол-во владельцев | INT |  |  |
| image | Путь фотографии | VARCHAR | 255 |  |
| asking\_price | Запрашиваемая цена | DECIMAL | 10, 2 |  |

Таблица 1.6.2 – Buyers (Покупатель)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Описание поля | Тип данных | Размер поля | Тип ключа |
| buyer\_id | ID покупателя | INT |  | PK |
| full\_name | ФИО | VARCHAR | 100 |  |

Продолжение таблицы 1.6.2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| phone\_number | Номер телефона | VARCHAR | 20 |  |
| city\_address | Город проживания | VARCHAR | 100 |  |
| email\_address | Эл. почта | VARCHAR | 100 |  |
| residential\_address | Адрес проживания | VARCHAR | 255 |  |
| seria\_passport | Серия паспорта | VARCHAR | 20 |  |
| number\_passport | Номер паспорта | VARCHAR | 20 |  |
| car\_brand\_requirements | Предпочтения к бренду | VARCHAR | 100 |  |
| technical\_requirements | Предпочтения к тех. харак | VARCHAR | 255 |  |
| budget | Бюджет | DECIMAL | 10, 2 |  |
| auto\_id | ID автомобиля | INT |  | FK |

Таблица 1.6.3 – Employee (Справочник сотрудников)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Описание поля | Тип данных | Размер поля | Тип ключа |
| employee\_id | ID сотрудника | INT |  | PK |
| first\_name | Имя | VARCHAR | 55 |  |
| second\_name | Фамилия | VARCHAR | 55 |  |
| third\_name | Отчество | VARCHAR | 55 |  |
| post\_id | ID должности | INT |  | FK |
| salary | Зарплата | DECIMAL | 10, 2 |  |
| image | Фото | VARCHAR | 255 |  |

Продолжение таблицы 1.6.3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| login | Логин | VARCHAR | 55 |  |
| password | Пароль | VARCHAR | 20 |  |

Таблица 1.6.4 – Sales (Справочник продажи)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Описание поля | Тип данных | Размер поля | Тип ключа |
| sale\_id | ID продажи | INT |  | PK |
| auto\_id | ID авто | INT |  | FK |
| buyer\_id | ID покупателя | INT |  | FK |
| price | Цена | DECIMAL | 10, 2 |  |
| warranty\_months | Гарантия | INT |  |  |
| sale\_date | Дата продажи | DATE |  |  |

Таблица 1.6.5 – Body (Справочник тип кузова)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Описание поля | Тип данных | Размер поля | Тип ключа |
| body\_id | ID кузова | INT |  | PK |
| body\_type | Тип кузова | VARCHAR | 55 |  |

Таблица 1.6.6 – Transmission (Справочник тип КПП)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Описание поля | Тип данных | Размер поля | Тип ключа |
| transmission\_id | ID трансмиссии | INT |  | PK |
| transmission\_type | Тип трансмиссии | VARCHAR | 55 |  |

Таблица 1.6.7 – Brand (Справочник бренды)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Описание поля | Тип данных | Размер поля | Тип ключа |
| brand\_id | ID бренда | INT |  | PK |
| brand\_name | Бренд | VARCHAR | 55 |  |

Таблица 1.6.8 – ConditionAuto (Справочник состояние авто)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Описание поля | Тип данных | Размер поля | Тип ключа |
| condition\_id | ID состояния | INT |  | PK |
| condition\_name | Состояние | VARCHAR | 55 |  |

Таблица 1.6.9 – StatusAuto (Справочник статус авто)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Описание поля | Тип данных | Размер поля | Тип ключа |
| status\_id | ID статуса | INT |  | PK |
| status\_name | Статус | VARCHAR | 55 |  |

Таблица 1.6.10 – Post (Справочник должность сотрудника)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Описание поля | Тип данных | Размер поля | Тип ключа |
| post\_id | ID должности | INT |  | PK |
| post\_name | Должность | VARCHAR | 55 |  |

1.7. Контрольный пример

Контрольный пример является ручным подсчётом задачи. По составленной программе обрабатываются исходные данные контрольного примера. Полученные результаты сравниваются с известными результатами контрольного примера. При несовпадении результатов производится поиск, исправление ошибок, и снова производится выполнение программы.

Входная информация контрольных примеров представлена в приложении А.

Выходные данные для контрольных примеров показаны в приложении Б.

1.8 Общие требования к программному продукту

Пользователи должны иметь базовые навыки пользования персональным компьютером.

Минимальные требования к техническому обеспечению программного продукта следующие:

* ОС: Windows 7 Service Pack1/ Windows 7 64Bit Service Pack1/ Windows 8.1 64Bit / Windows 10 64Bit / Mac OS / Linux OS;
* процессор: 2.40 Ггц (четырехъядерный) / AMD Phenom 9850 (четырехъядерный) @ 2.5 Ггц;
* оперативная память: 4гб (Win 7/Win8 и выше);
* видеокарта: NVIDIA 9800 GT c 512 МБ видеопамяти/ AMD HD 4870 с 1 Гб видеопамяти (DX 9, 10, 10.1);
* жесткий диск: 10 гигабайт свободного места;
* Microsoft DirectX версия 9.0c;

Функциональные возможности программного продукта:

* приложение должно формировать и отображать выходные данные пользователю;
* в приложении должен быть обеспечен просмотр таблиц (справочников) базы данных с возможность добавления, редактирования, удаления данных.

Требования к надежности:

* приложение должно обрабатывать ошибочные действия пользователя и сообщать ему об этом;
* приложение должно обеспечивать контроль входной и выходной информации.

Требования к информационной и программной совместимости: обеспечить работу приложения с таблицами СУБД MySQL.

2 Экспериментальный раздел

2.1 Описание программы

При написании программы «AutoMir» использовались редакторы кода Visual Studio с использованием системы управления базами данных MySQL.

Программа написана на языке C#. Так же, использовался фреймворк WPF для визуальной части.

Схема взаимодействия модулей программы представлена на рисунке 2.1.1. Описание модулей и методов представлено в таблице 2.1.1.

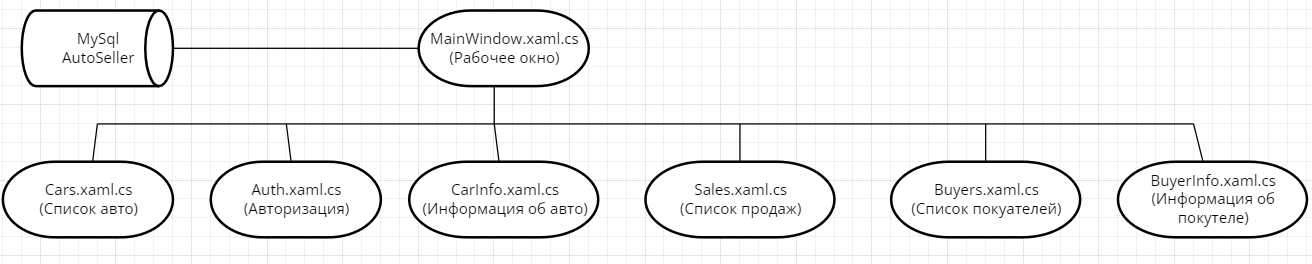


Рисунок 2.1.1 – Схема взаимодействия модулей

Таблица 2.1.1. – Описание модулей

|  |  |
| --- | --- |
| Методы | Назначение |
| 1 | 2 |
| MainWindow.xaml.cs | |
| public MainWindow() | Переход на страницу входа |
| private void CarsClick(object sender, RoutedEventArgs e) | Переход к странице Автомобилей |
| private void Sales\_Click(object sender, RoutedEventArgs e) | Переход к странице Продажи |

Продолжение таблицы 2.1.1

|  |  |
| --- | --- |
| 1 | 2 |
| private void Auth\_Click(object sender, RoutedEventArgs e) | Переход к странице авторизации |
| private void Buyers\_Click(object sender, RoutedEventArgs e) | Переход к странице Покупатели |

|  |  |
| --- | --- |
| Cars.xaml.cs | |
| public Cars(Frame frame) | Переход на Frame |
| private void addcar\_Click(object sender, RoutedEventArgs e) | Переход на страницу добавления автомобилей |
| private void search\_TextChanged(object sender, TextChangedEventArgs e) | Метод для поиска авто по модели |
| private void brandSel\_SelectionChanged(object sender, SelectionChangedEventArgs e) | Метод для сортировки авто по бренду |
| private void kuzSel\_SelectionChanged(object sender, SelectionChangedEventArgs e) | Метод для сортировки авто по типу кузова |
| Auth.xaml.cs | |
| public Auth(Frame frame) | Переход на Frame |
| private void Auth\_Click(object sender, RoutedEventArgs e) | Метод для авторизации сотрудника |
| CarInfo.xaml.cs | |
| public CarInfo(int carId, Frame frame) | Переход на Frame c ID автомобиля |
| private void CarDataLoad() | Метод для загрузки в TextBox данные об авто |

Продолжение таблицы 2.1.1

|  |  |
| --- | --- |
| 1 | 2 |
| private void SaveChangesButton\_Click(object sender, RoutedEventArgs e) | Метод для сохранение изменений |
| Sales.xaml.cs | |
| public Sales(Frame frame) | Переход на Frame |
| private void CreateSale\_Click(object sender, RoutedEventArgs e) | Переход на страницу добавления продажи |
| private void search\_TextChanged(object sender, TextChangedEventArgs e) | Метод для поиска продаж по ФИО покупателя |
| private void datesale\_MouseDown(object sender, MouseButtonEventArgs e) | Сортировка по дате продажи |
| Buyers.xaml.cs | |
| public Buyers(Frame frame) | Переход на Frame |
| private void addBuy\_Click(object sender, RoutedEventArgs e) | Переход на страницу добавления покупателя |
| private void search\_TextChanged(object sender, TextChangedEventArgs e) | Метод для поиска покупателей по ФИО |
| BuyersInfo.xaml.cs | |
| public BuyerInfo(int buyerID, Frame frame) | Переход на Frame c ID автомобиля |
| private void BuyerDataLoad() | Метод для загрузки в TextBox данные об покупателе |
| private void ChangeBuyer(object sender, RoutedEventArgs e) | Метод для сохранение изменений |

Код программы представлен в приложении Г.

2.2 Протокол тестирования программного продукта

В протоколе тестирования отражаются:

- тестирование на корректных данных;

- тестирование на некорректных данных;

- тестирование продукта на данных контрольного примера.

Тестирование на сохранение данных об автомобиле при корректных данных, ожидаемое сообщение «Автомобиль добавлен» (рисунок 2.2.1).

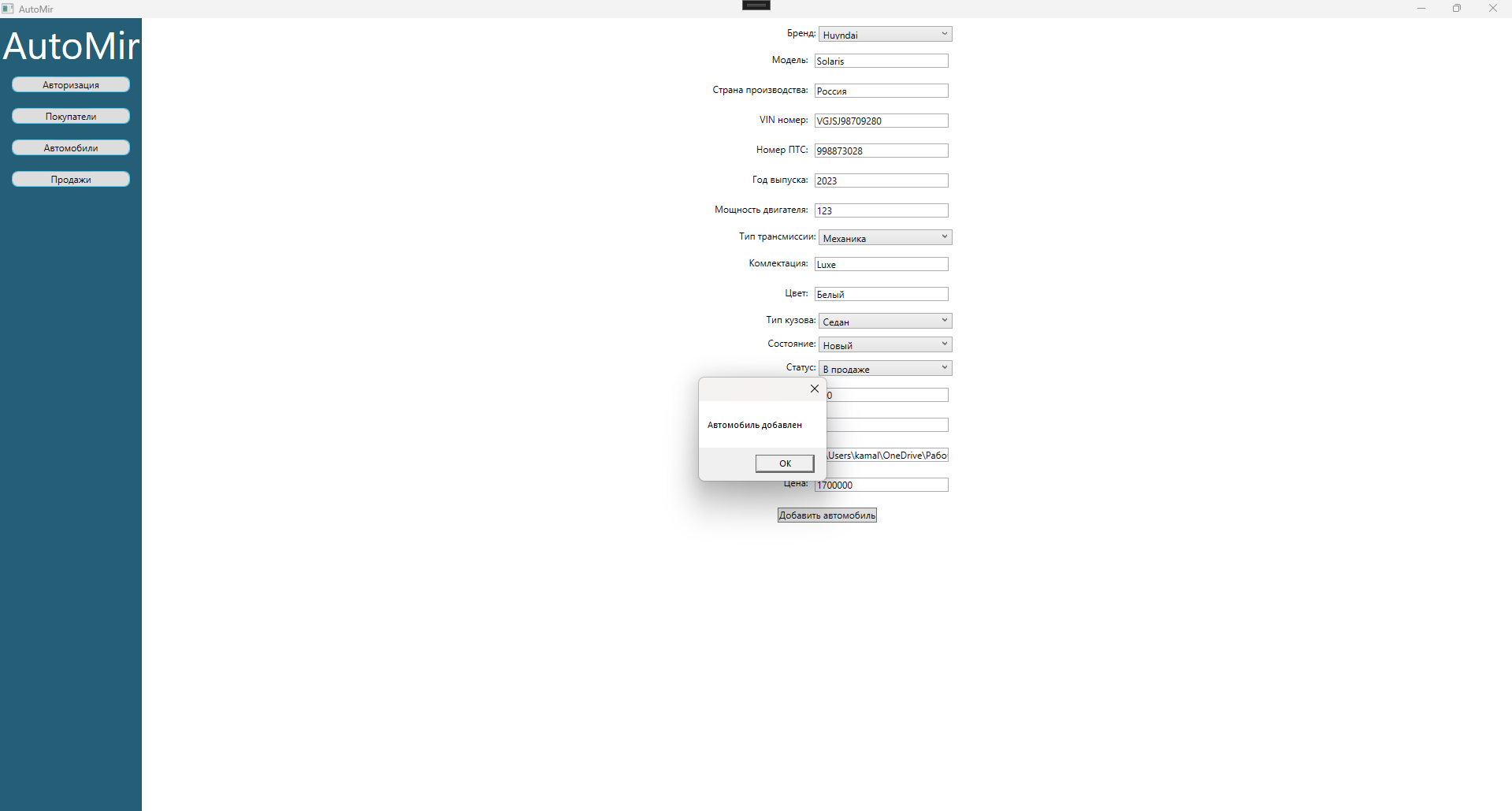


Рисунок 2.2.1 – Сообщение об успешном добавлении автомобиля в базу

Тестирование просмотр и редактирование данных автомобиля, ожидаемое сообщение «Изменение внесены». Чтобы сохранить изменения, нажмите на кнопку «Сохранить изменения» (рисунок 2.2.2).

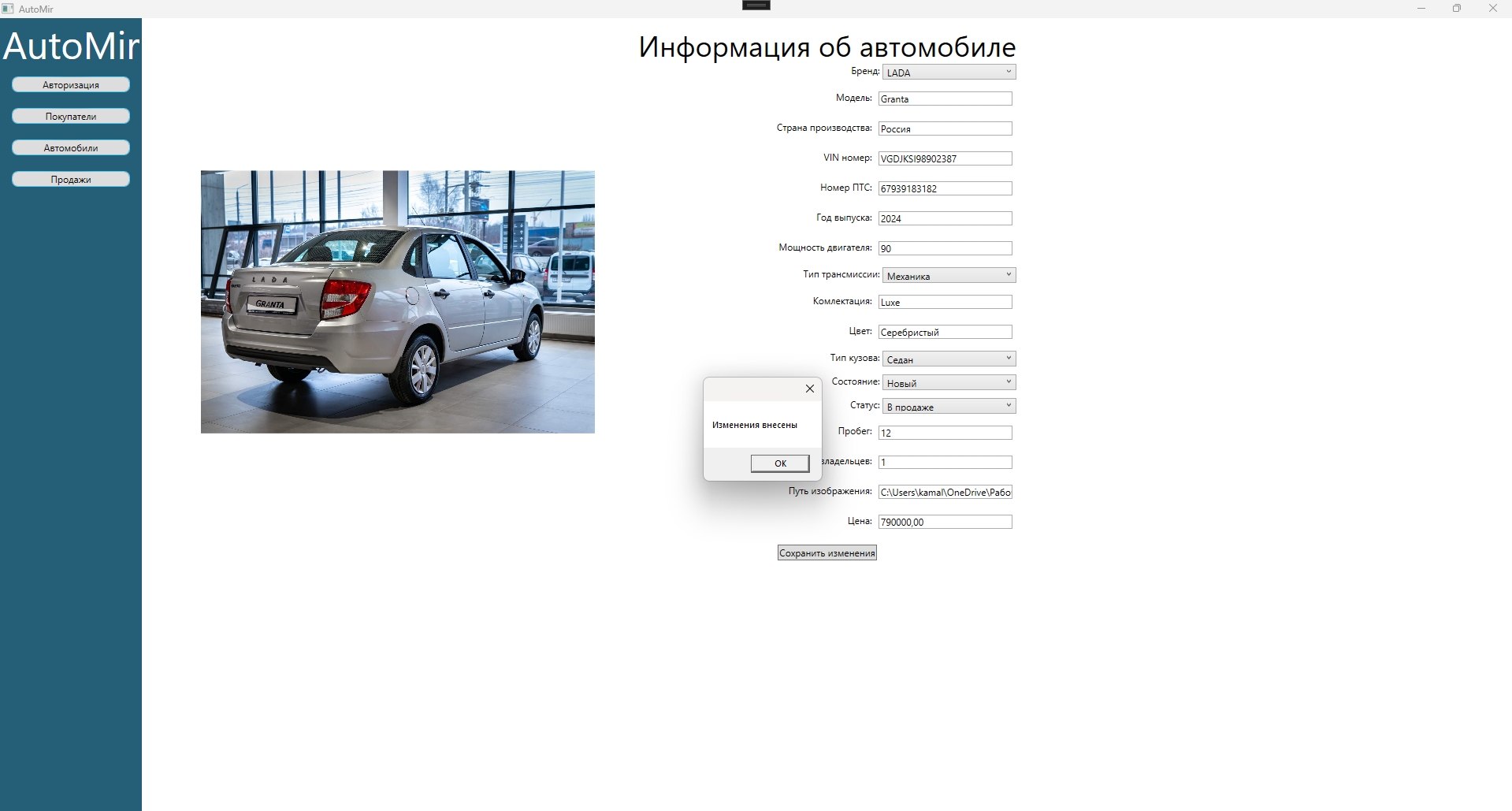


Рисунок 2.2.2 – Сообщение об успешном сохранении изменений данных автомобиля

Тестирование на некорректных данных при авторизации, ожидаемое сообщение «Неверный логин или пароль». Чтобы проверить, нажмите на кнопку «Войти» (рисунок 2.2.3).

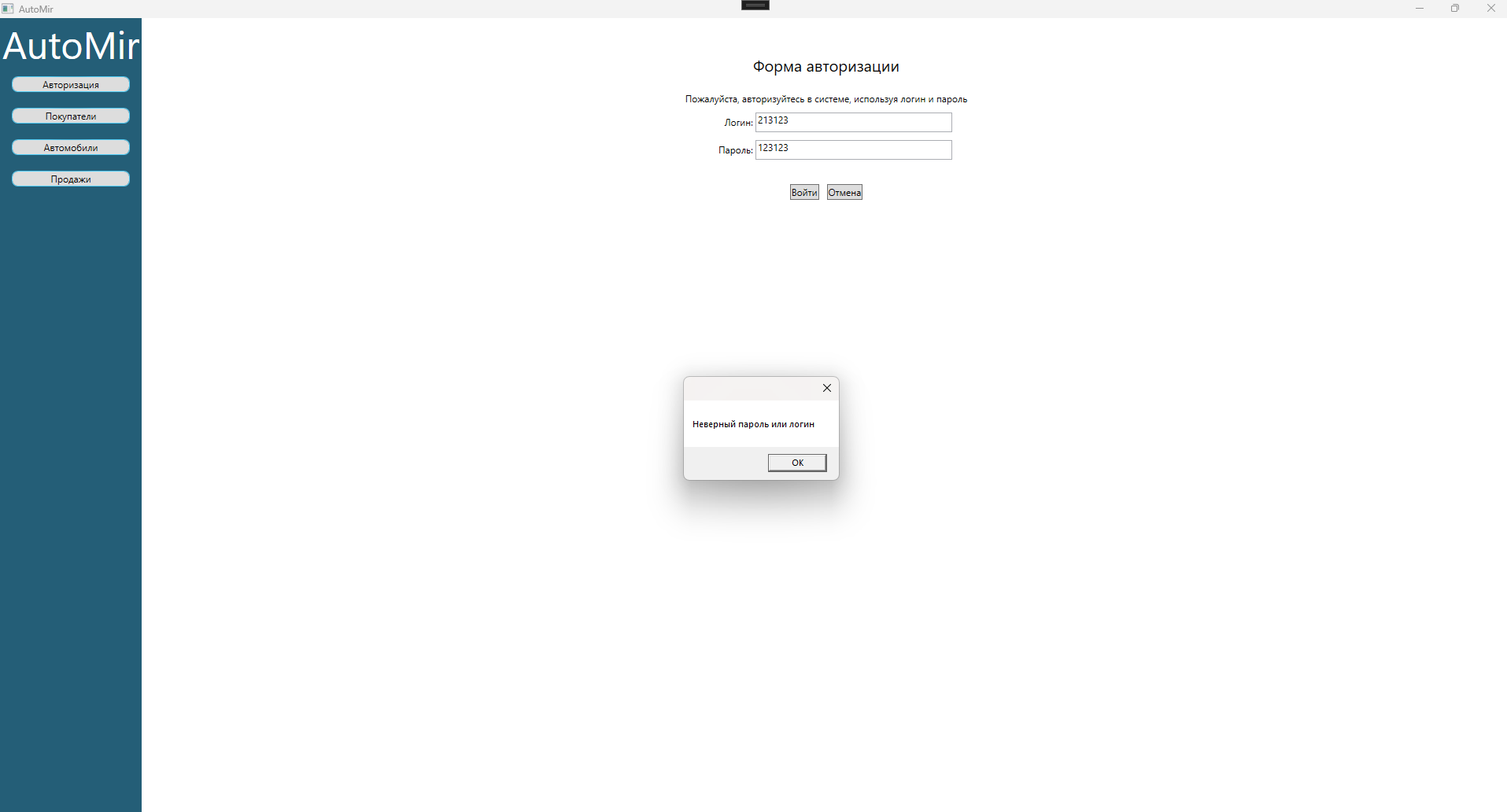


Рисунок 2.2.3 – Сообщение об неверной авторизации

Тестирование на корректных данных при авторизации, ожидаемое сообщение «Вы вошли как – (Должность сотрудника)». Чтобы проверить, нажмите на кнопку «Войти» (рисунок 2.2.4).

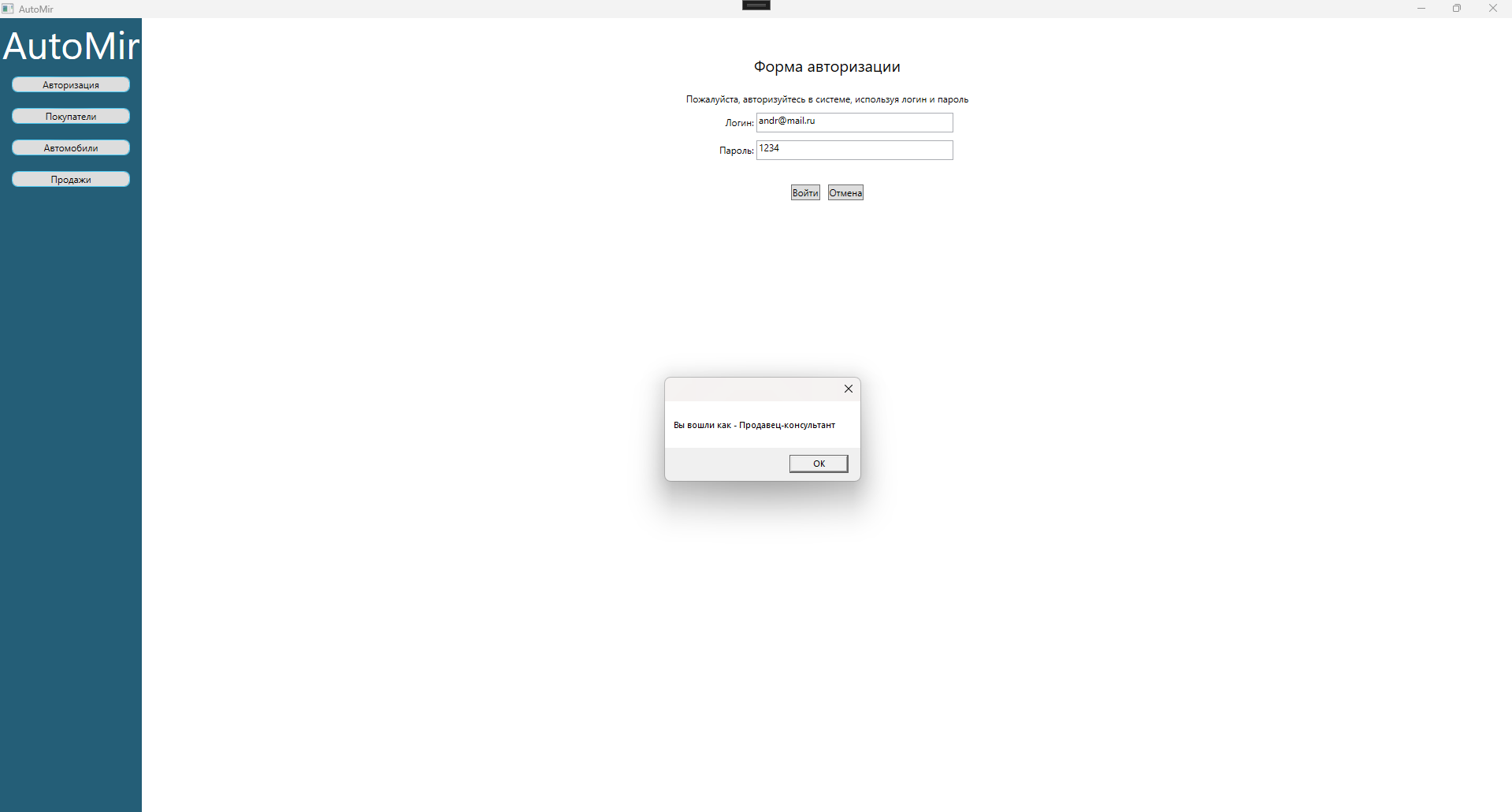


Рисунок 2.2.4 – Сообщение об успешной авторизации

Тестирование на корректных данных при добавлении продажи, ожидаемое сообщение «Документ сформирован и запись о продаже добавлена в базу данных». Чтобы проверить, нажмите на кнопку «Оформить продажу» (рисунок 2.2.5) (рисунок 2.2.6).

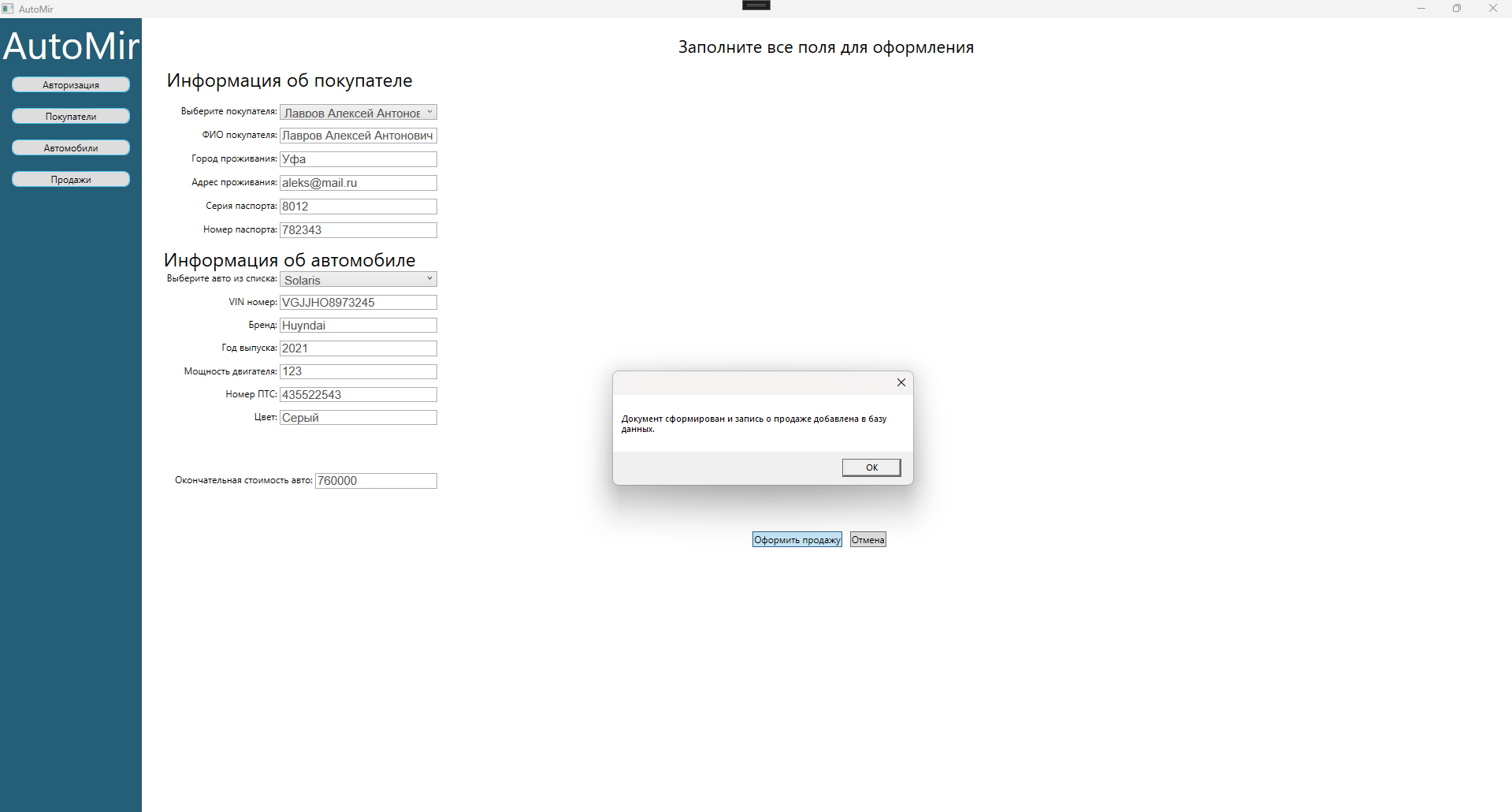


Рисунок 2.2.5 – Сообщение об успешном оформлении продажи

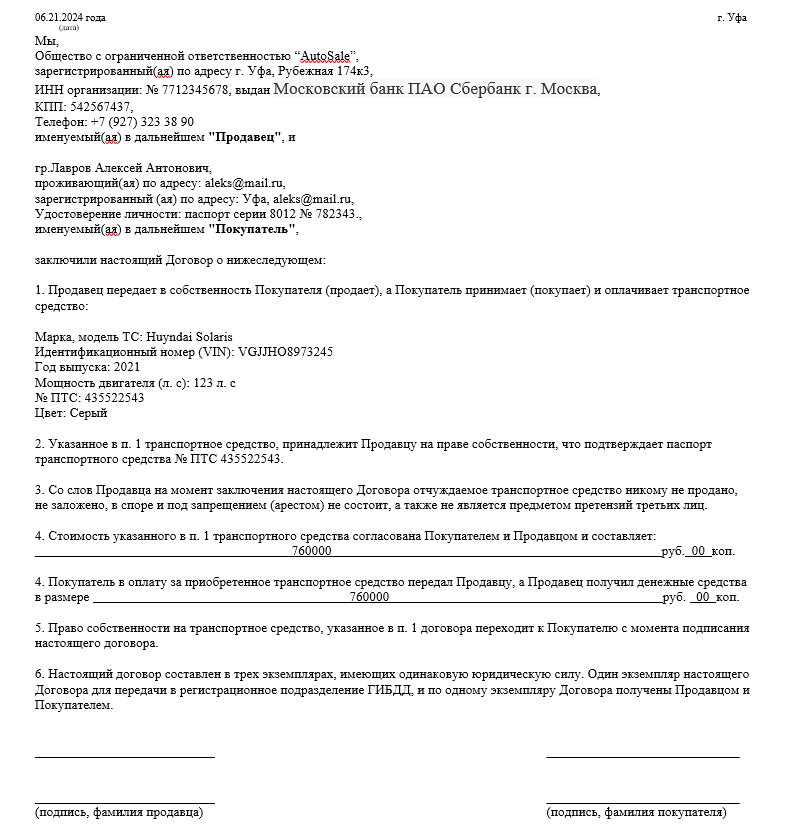


Рисунок 2.2.6 – Сформированный договор купли-продаже в ворд файле после оформления продажи

2.3 Руководство пользователя

Назначение системы

Программа «AutoMir» предназначена для упрощения оформления продажей авто, поиск по параметрам автомобиля для клиентов, а также для составления договора.

Основной целью данной информационной системы является просмотр списков автомобилях, сформированных по различным критериям, и информации о них.

Приложение «AutoMir» предназначено для пользователей, имеющих как минимум первоначальные навыки работы с графической операционной системой, на которой будет запускаться данная программа.

Условия применения системы

Программное обеспечение разрабатывается для персональной вычислительной техники со следующими характеристиками:

* Microsoft Windows 7 / 8 / 10;
* процессор 1 ГГц;
* 128 МБ ОЗУ;
* 60 МБ свободного пространства на диске;
* разрешение экрана монитора не менее 1920 × 1080;
* дополнительное оборудование
* принтер.

Подготовка системы к работе

Для запуска программы необходимо запустить приложение AutoMir.exe из каталога, в котором установлен данный программный продукт. После этого открывается окно авторизации для входа в программу (рисунок 2.3.1).

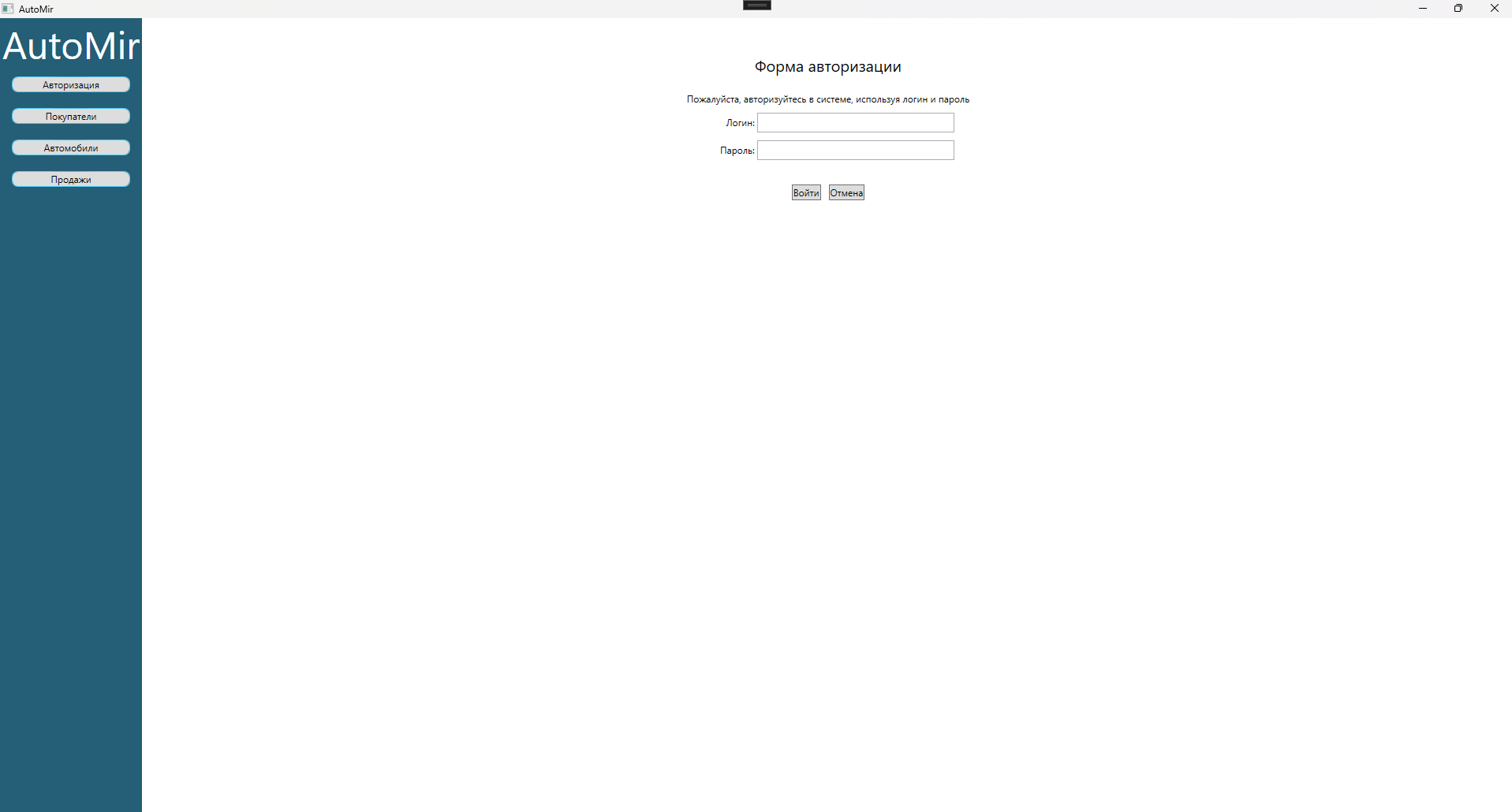


Рисунок 2.3.1 – Окно авторизации для входа в программу

Описание операций

После ввода правильного логина и пароля осуществляется вход в приложение. После авторизации открывается страница со списком автомобилей, а также будут доступны другие вкладки на панели (Автомобили, Покупатели, Продажи) (рисунок 2.3.2).

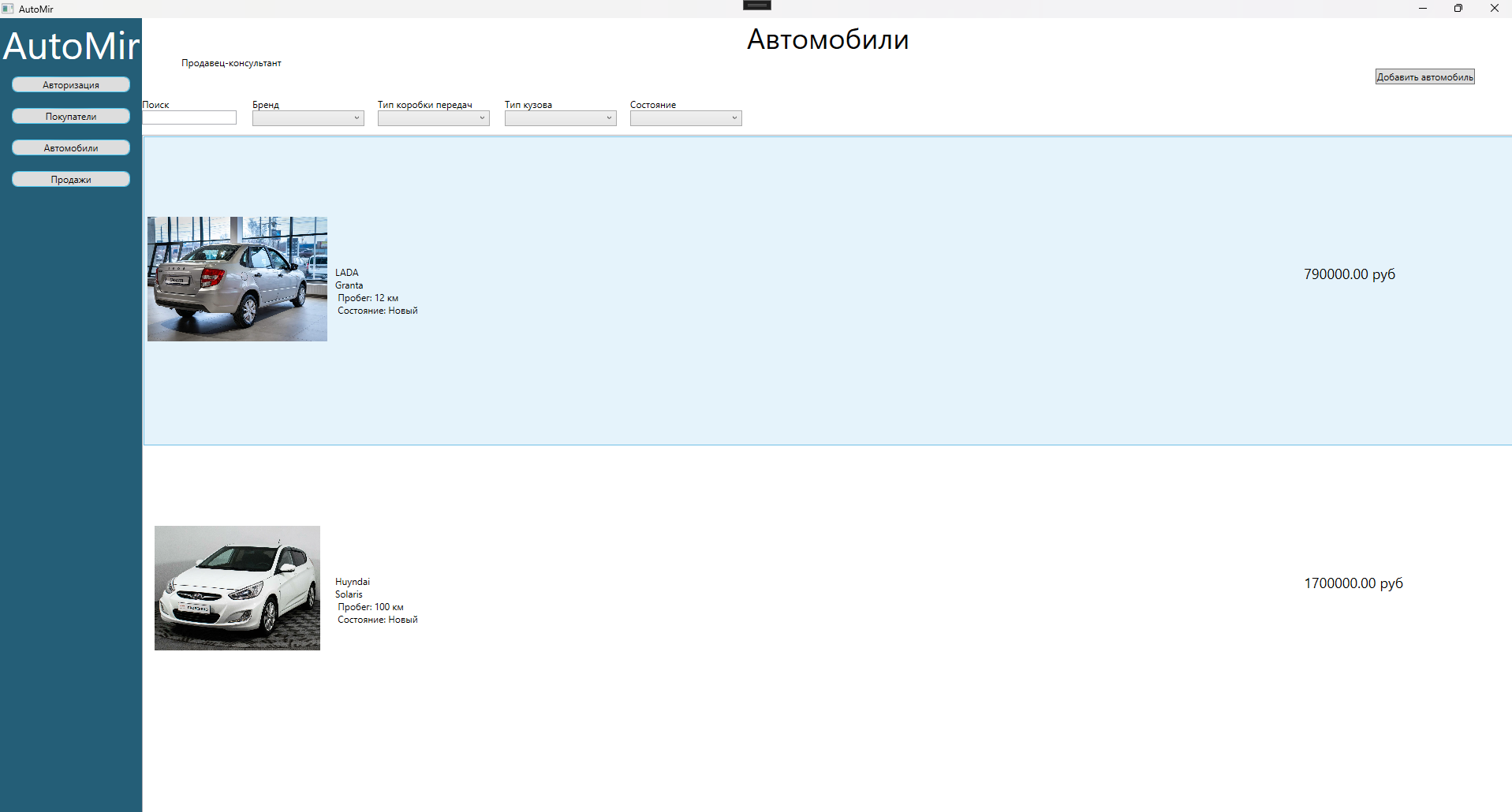


Рисунок 2.3.2 – Окно с автомобилями после авторизации

Для добавления автомобиля в базу данных нужно нажать кнопку «Добавить автомобиль», после чего вас перенаправит на страничку добавления (рисунок 2.3.3).

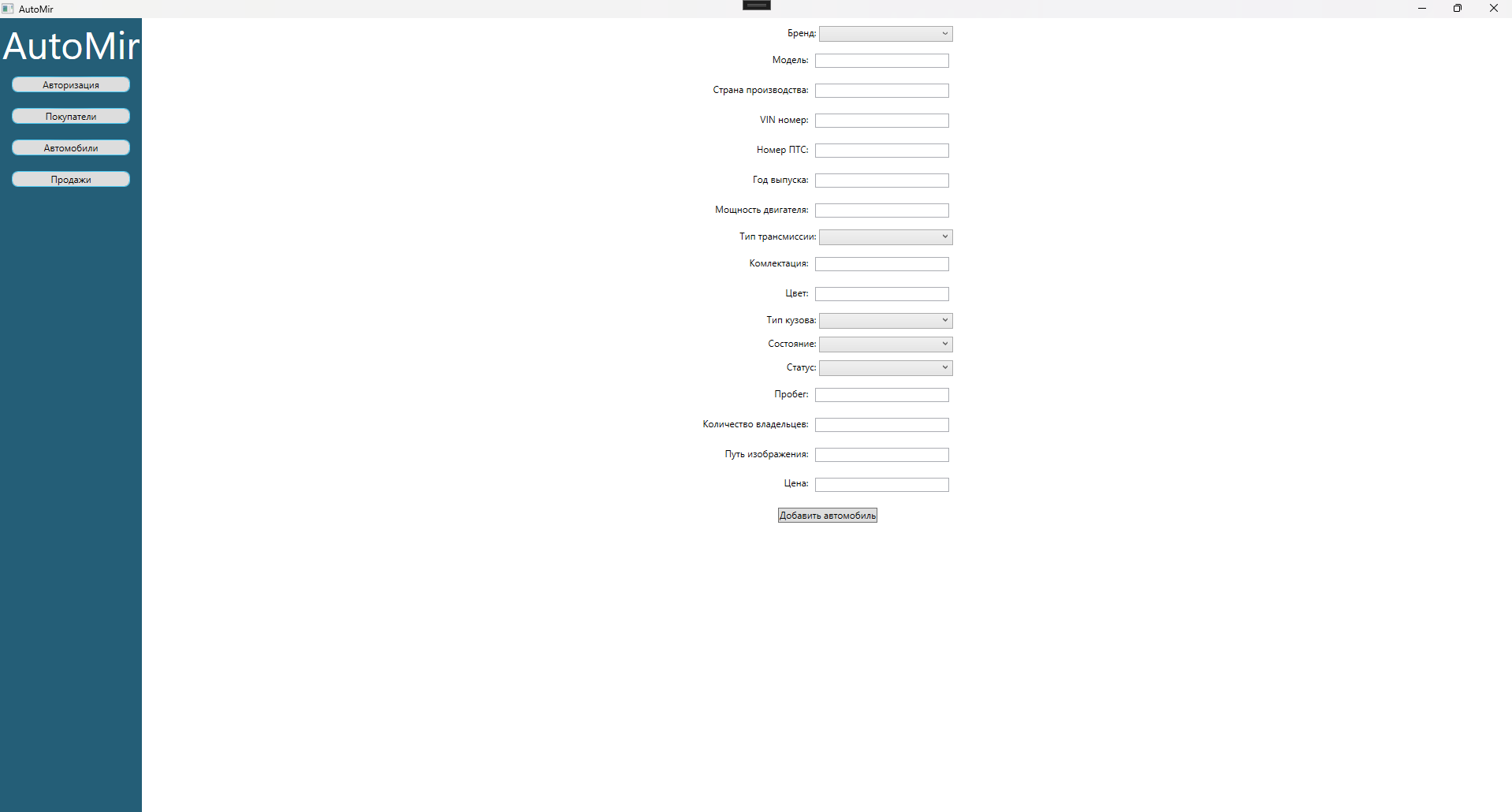


Рисунок 2.3.3 – Окно добавления авто

Для редактирования данных автомобиля вам нужно нажать левой кнопкой мыши на сам автомобиль, который вы хотите отредактировать (рисунок 2.3.4).

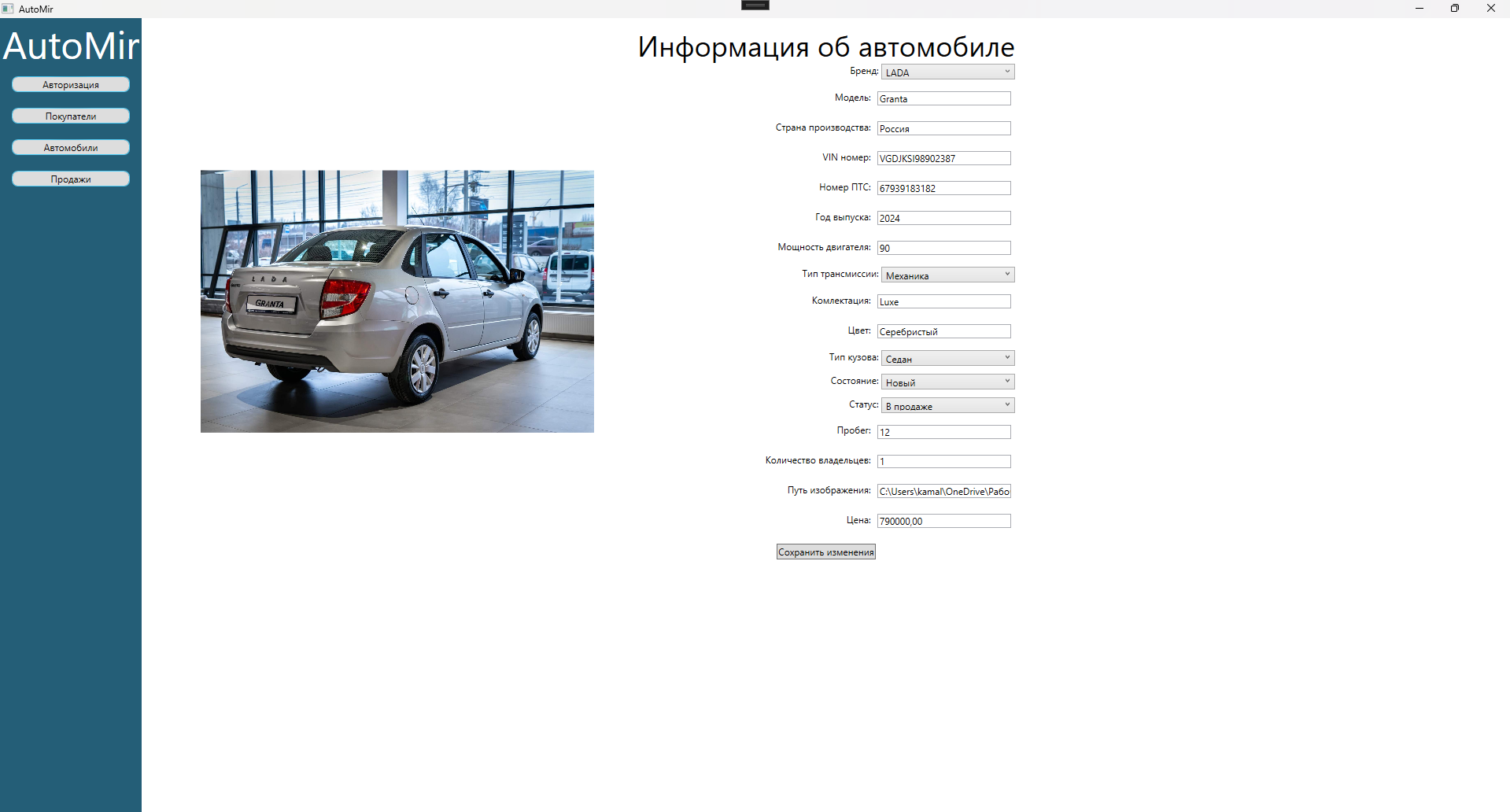


Рисунок 2.3.4 – Окно редактирования авто

Чтобы посмотреть список покупателей нажмите на кнопку на боковой панели «Покупатели» (рисунок 2.3.5).

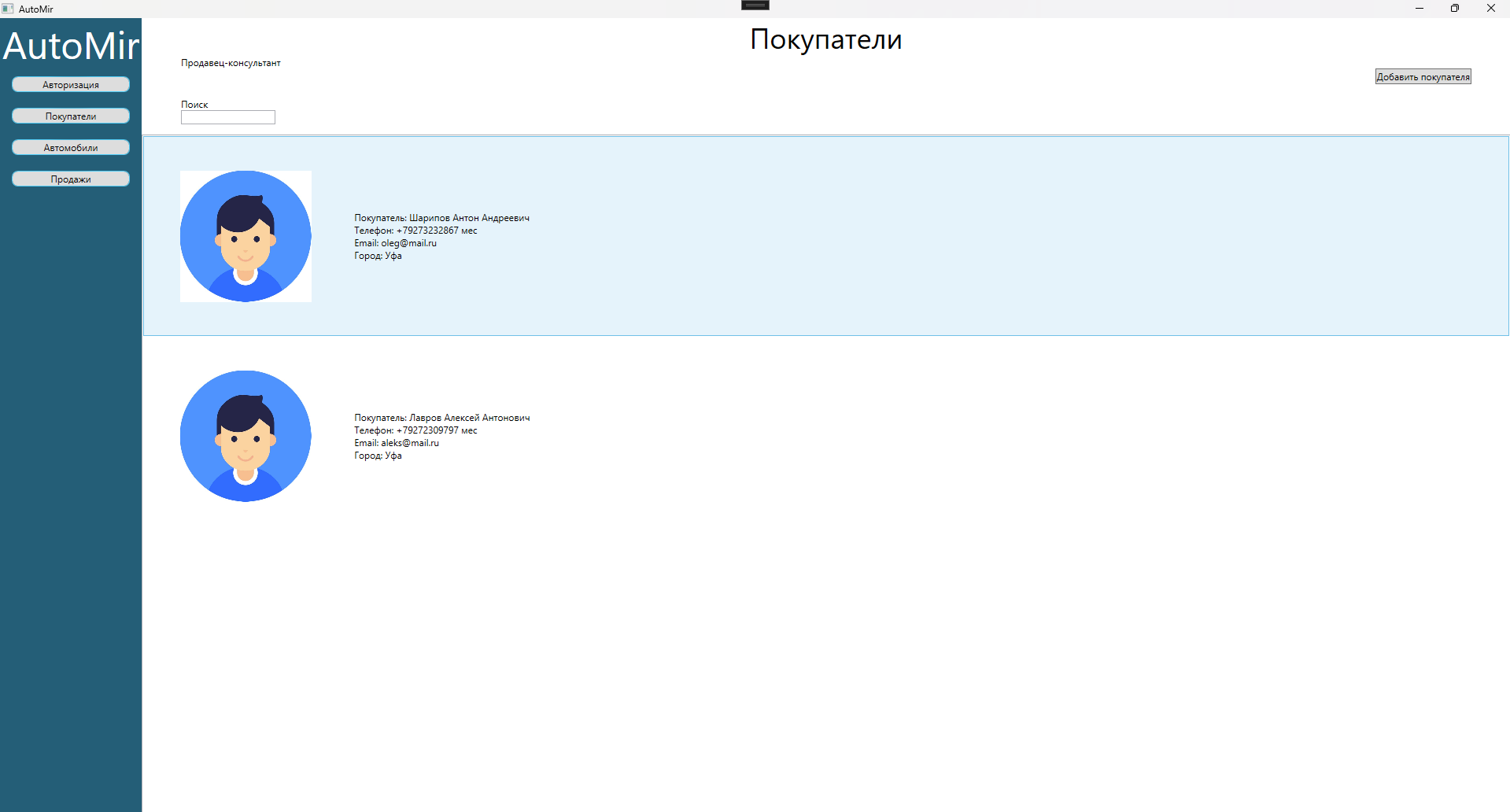


Рисунок 2.3.5 – Окно списка покупателей

Чтобы добавить покупателя, нажмите кнопку на боковой панели «Добавить покупателя» (рисунок 2.3.6).

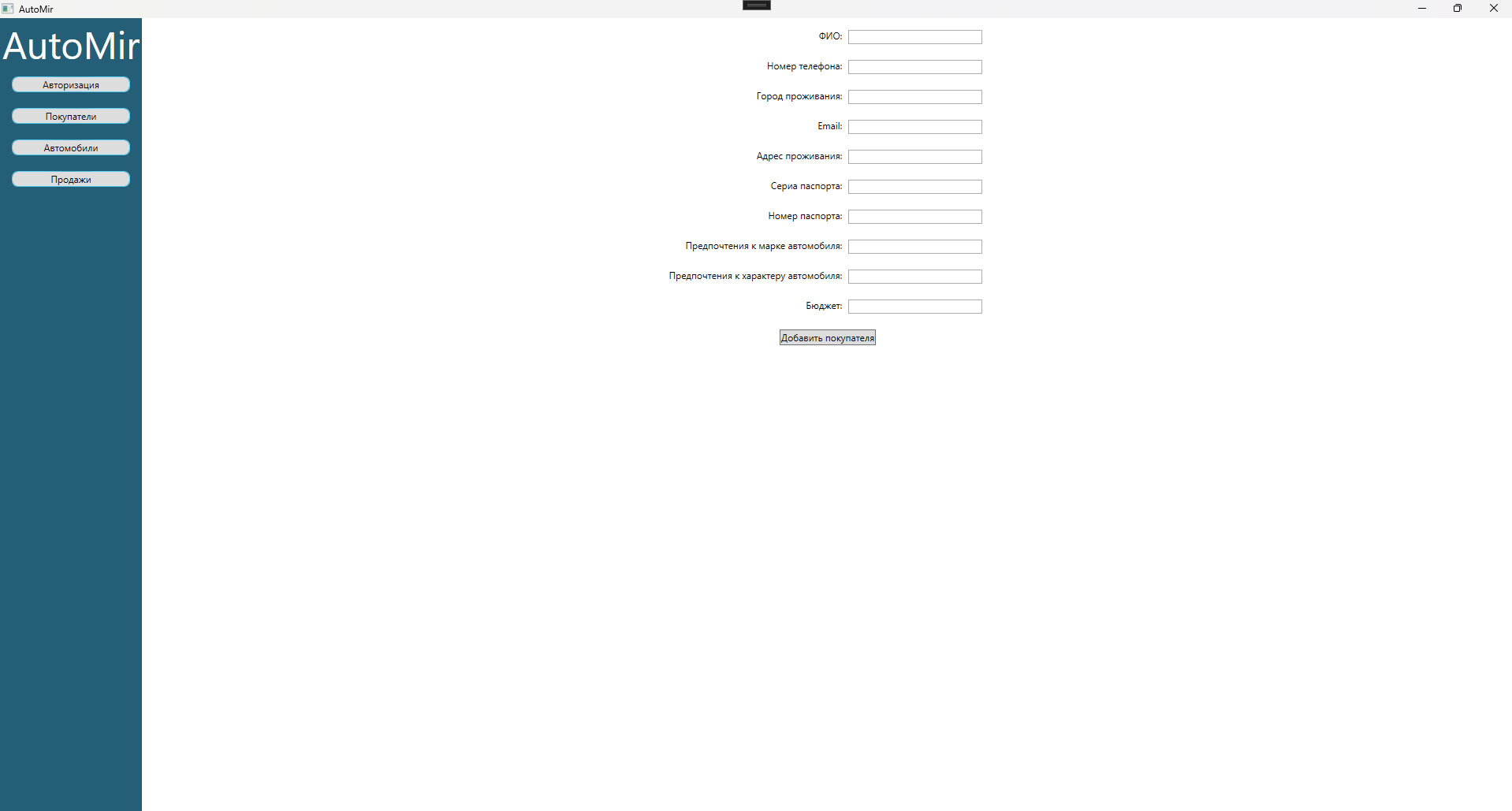


Рисунок 2.3.6 – Окно добавления покупателей

Для редактирования данных покупателя вам нужно нажать левой кнопкой мыши на самого покупателя, который вы хотите отредактировать (рисунок 2.3.7).

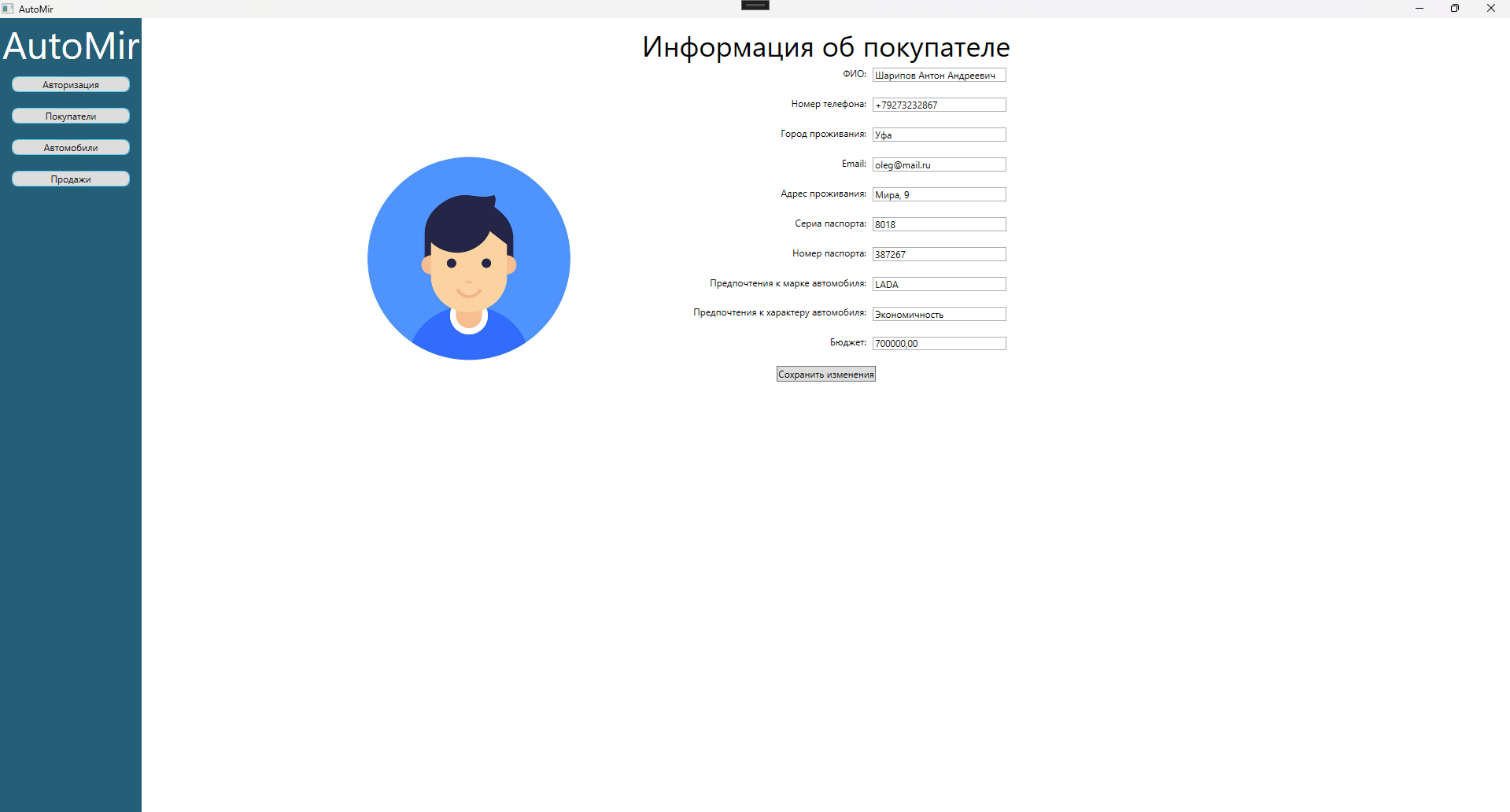


Рисунок 2.3.7 – Окно редактирования покупателя

Для перехода в раздел продажи нажмите на кнопку «Продажи» левой кнопкой мыши, только при условии того что вы авторизованный пользователь (рисунок 2.3.8).

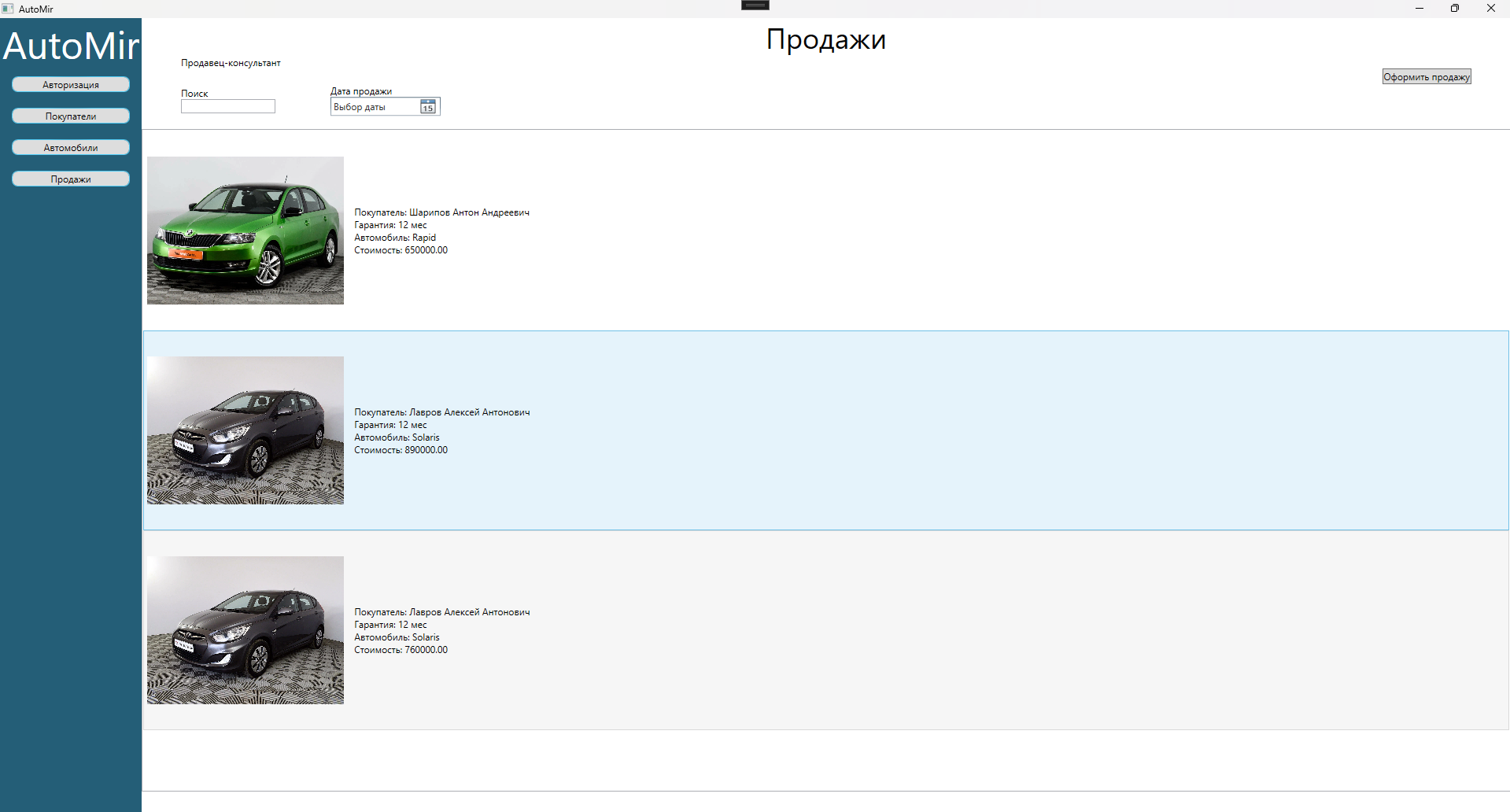


Рисунок 2.3.8 – Окно список продаж

Для того чтобы оформить продажу, нажмите на кнопку на верхней панели «Оформить продажу», выберите из выпадающего списка покупателя и автомобиль (рисунок 2.3.9).

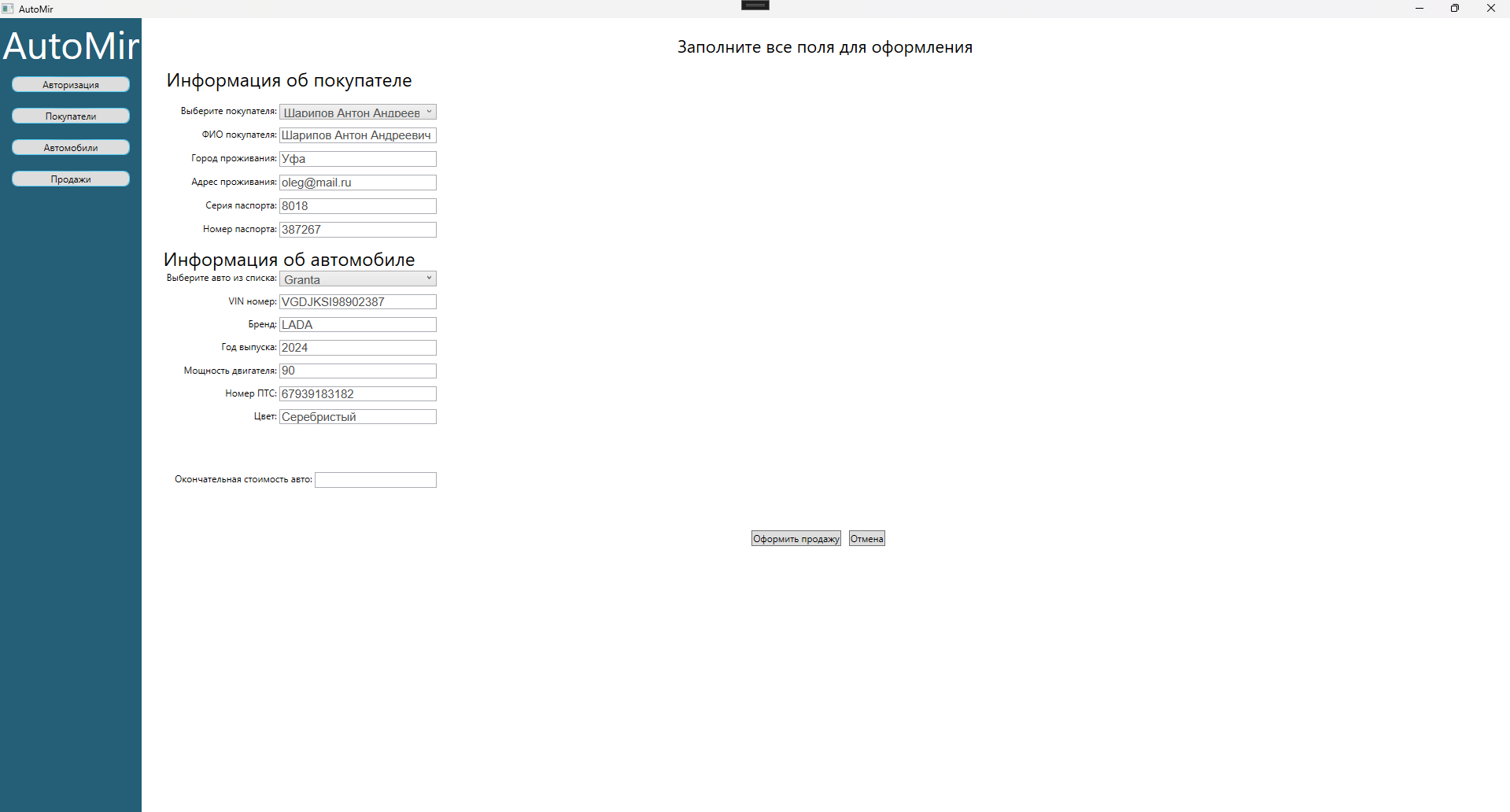


Рисунок 2.3.9 – Окно оформления продажи

После оформления продажи из списка продаваемых авто пропадает та машина, которую только что продали (рисунок 2.3.10).

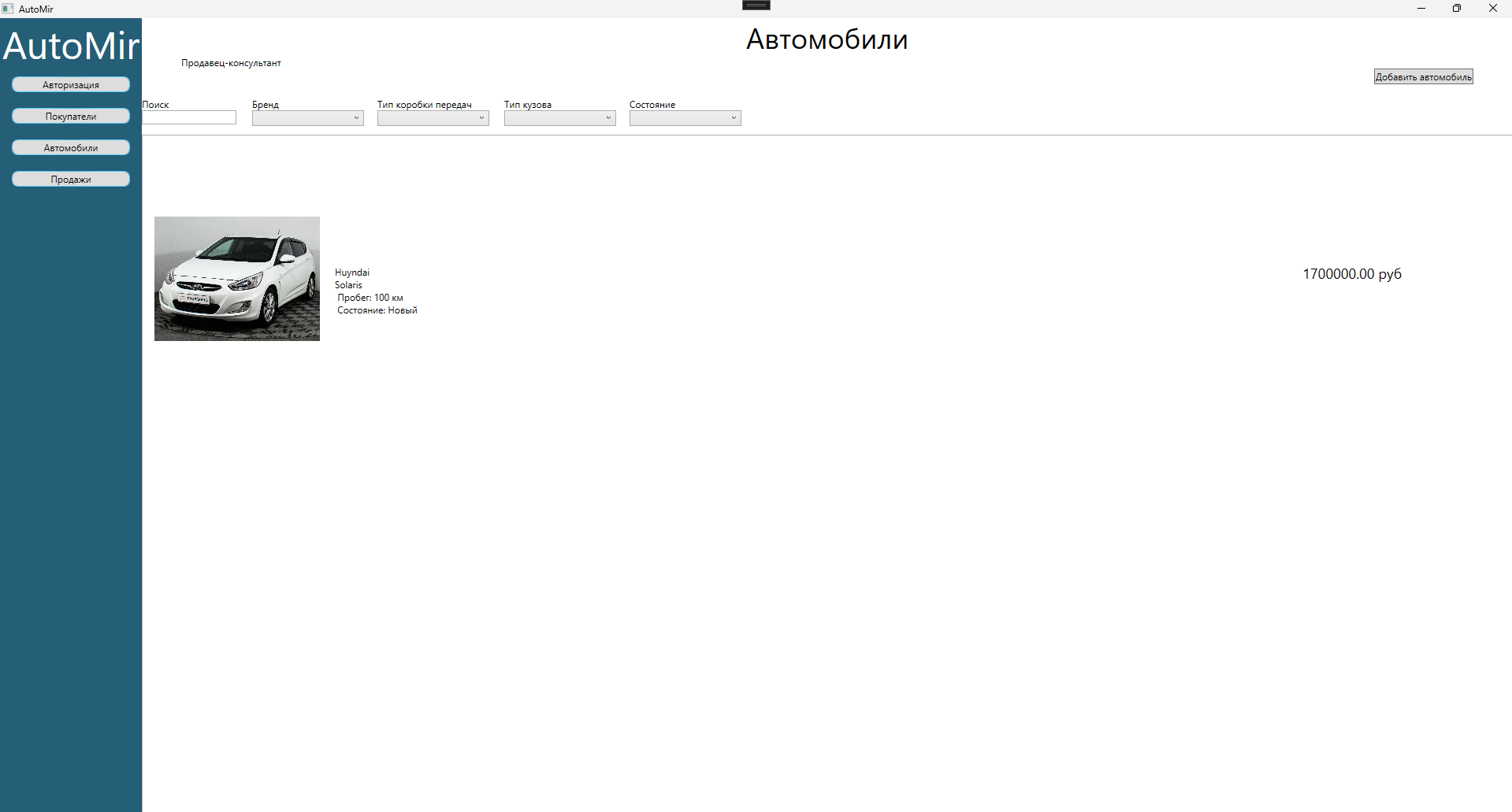


Рисунок 2.3.10 – Окно со списком автомобилей после продажи

ЗАКЛЮЧЕНИЕ

В процессе выполнения курсового проекта были разработаны структура и алгоритм работы программы «AutoSale».

Результатом работы стало создание программы для сокращения времени сотрудников автосалон в создание процесса продаж автомобилей.

Программ AutoMir написано на языках C# с использованием системы управления базами данных MySQL, а также с использованием фреймворка WPF, предназначено для работы на операционной системе Windows 10 и выше.

Были проведены опытная эксплуатация и отладочное тестирование программы. После этого было написано руководство пользователя.

С помощью приложения на основании данных контрольного примера были получены результаты, которые полностью совпадают с выходной информацией контрольного примера.

Входные данные контрольного примера

Таблица А.1 – Справочник сотрудников

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID сотрудника | Имя | Фамилия | login | password |
| 1 | Максим | Камалов | max@mail.ru | 1234 |
| 2 | Андрей | Ушаков | [andr@mail.ru](mailto:andr@mail.ru) | 1234 |

Таблица А.2 – Справочник покутелей

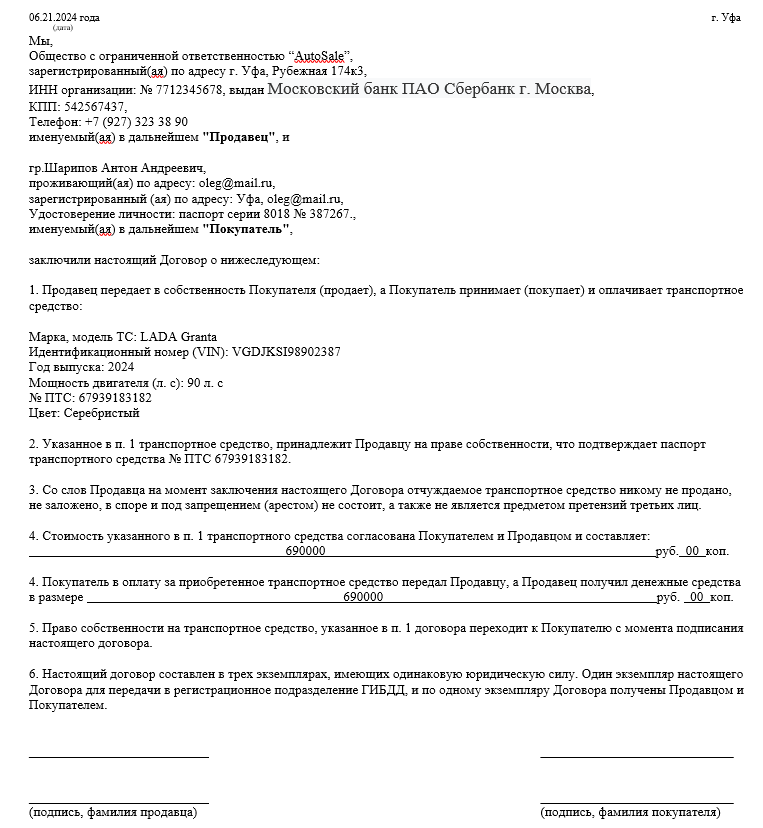
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ID покупателя | ФИО | Номер телефона | Город | Email | Адрес проживания | Серия паспорта | Номер паспорта |
| 1 | Шарипов Антон Андреевич | +79273232867 | Уфа | oleg@mail.ru | Мира, 9 | 8018 | 387267 |
| 2 | Лавров Алексей Антонович | +79272309797 | Уфа | aleks@mail.ru | Первомайска, 29 | 8012 | 782343 |

Таблица А.3 – Справочник автомобилей

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ID авто | ID бренд | Модель | Страна | VIN | Номер ПТС | Год | Мощность двигателя |
| 2 | 1 | Granta | Россия | VGDJKSI98902387 | 67939183182 | 2024 | 90 |
| 3 | 3 | Solaris | Корея | VGJJHO8973245 | 435522543 | 2021 | 123 |
| 4 | 6 | Rapid | Чехия | VHHFDG213123412 | 23242412 | 2020 | 108 |

Выходные данные контрольного примера

Рисунок Б.1 – Договор купли-продажи



MainWindow.xaml.cs

using AutoMir.Model;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace AutoMir

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

MainFrame.Navigate(new Auth(MainFrame));

}

private void CarsClick(object sender, RoutedEventArgs e)

{

Employee user1 = Security.user;

if (user1 == null)

{

MessageBox.Show("Вы не авторизированы!");

}

else

{

MainFrame.Navigate(new Cars(MainFrame));

}

}

private void Sales\_Click(object sender, RoutedEventArgs e)

{

Employee user1 = Security.user;

if (user1 == null)

{

MessageBox.Show("Вы не авторизированы!");

}

else

{

MainFrame.Navigate(new Sales(MainFrame));

}

}

private void Auth\_Click(object sender, RoutedEventArgs e)

{

MainFrame.Navigate(new Auth(MainFrame));

}

private void Buyers\_Click(object sender, RoutedEventArgs e)

{

Employee user1 = Security.user;

if (user1 == null)

{

MessageBox.Show("Вы не авторизированы!");

}

else

{

MainFrame.Navigate(new Buyers(MainFrame));

}

}

}

}

AddBuyer.xaml.cs

using AutoMir.Model.Data;

using AutoMir.Model;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace AutoMir

{

/// <summary>

/// Логика взаимодействия для AddBuyer.xaml

/// </summary>

public partial class AddBuyer : Page

{

private Frame frame;

public AddBuyer(Frame frame)

{

InitializeComponent();

this.frame = frame;

}

private void AddBuyButton\_Click(object sender, RoutedEventArgs e)

{

using (AutosallerContext context = new AutosallerContext())

{

var buyer = new Buyer

{

FullName = fio.Text,

PhoneNumber = phonenumber.Text,

CityAddress = city.Text,

EmailAddress = email.Text,

ResidentialAddress = adress.Text,

SeriaPassport = seria.Text,

NumberPassport = nomer.Text,

CarBrandRequirements = req.Text,

TechnicalRequirements = tech.Text,

Budget = decimal.TryParse(bud.Text, out decimal budget) ? (decimal?)budget : null

};

context.Buyers.Add(buyer);

context.SaveChanges();

}

MessageBox.Show("Покупатель добавлен");

}

}

}

AddCar.xaml.cs

using AutoMir.Model;

using AutoMir.Model.Data;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace AutoMir

{

/// <summary>

/// Логика взаимодействия для AddCar.xaml

/// </summary>

public partial class AddCar : Page

{

private Frame frame;

public AddCar(Frame frame)

{

InitializeComponent();

this.frame = frame;

using (AutosallerContext context = new AutosallerContext())

{

List<string> items = context.Brands.Select(p => p.BrandName).ToList();

BrandIdTextBox.ItemsSource = items;

List<string> items2 = context.Transmissions.Select(p => p.TransmissionType).ToList();

TransmissionIdTextBox.ItemsSource = items2;

List<string> items3 = context.Bodies.Select(p => p.BodyType).ToList();

BodyIdTextBox.ItemsSource = items3;

List<string> items4 = context.Conditionautos.Select(p => p.ConditionName).ToList();

ConditionIdTextBox.ItemsSource = items4;

List<string> items5 = context.Statusautos.Select(p => p.StatusName).ToList();

StatusIdTextBox.ItemsSource = items5;

}

}

private void AddCarButton\_Click(object sender, RoutedEventArgs e)

{

using (AutosallerContext context = new AutosallerContext())

{

var car = new Car

{

BrandId = context.Brands.FirstOrDefault(b => b.BrandName == BrandIdTextBox.SelectedItem.ToString())?.BrandId,

Model = ModelTextBox.Text,

CountryOfProduction = CountryOfProductionTextBox.Text,

VinNumber = VinNumberTextBox.Text,

PtsNumber = PtsNumberTextBox.Text,

YearOfManufacture = int.TryParse(YearOfManufactureTextBox.Text, out int yearOfManufacture) ? (int?)yearOfManufacture : null,

EnginePower = EnginePowerTextBox.Text,

TransmissionId = context.Transmissions.FirstOrDefault(t => t.TransmissionType == TransmissionIdTextBox.SelectedItem.ToString())?.TransmissionId,

Equipment = EquipmentTextBox.Text,

Color = ColorTextBox.Text,

BodyId = context.Bodies.FirstOrDefault(b => b.BodyType == BodyIdTextBox.SelectedItem.ToString())?.BodyId,

ConditionId = context.Conditionautos.FirstOrDefault(c => c.ConditionName == ConditionIdTextBox.SelectedItem.ToString())?.ConditionId,

StatusId = context.Statusautos.FirstOrDefault(s => s.StatusName == StatusIdTextBox.SelectedItem.ToString())?.StatusId,

Probeg = int.TryParse(ProbegTextBox.Text, out int probeg) ? (int?)probeg : null,

NumberOfOwners = int.TryParse(NumberOfOwnersTextBox.Text, out int numberOfOwners) ? (int?)numberOfOwners : null,

Image = ImageTextBox.Text,

AskingPrice = decimal.TryParse(AskingPriceTextBox.Text, out decimal askingPrice) ? (decimal?)askingPrice : null

};

context.Cars.Add(car);

context.SaveChanges();

}

MessageBox.Show("Автомобиль добавлен");

frame.Navigate(new Cars(frame));

}

}

}

AddSale.xaml.cs

using AutoMir.Model;

using AutoMir.Model.Data;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace AutoMir

{

/// <summary>

/// Логика взаимодействия для AddSale.xaml

/// </summary>

public partial class AddSale : Page

{

private Frame frame;

public AddSale(Frame frame)

{

InitializeComponent();

this.frame = frame;

using (AutosallerContext context = new AutosallerContext())

{

List<string> items = context.Cars.Where(v => v.StatusId == 1).Select(p => p.Model).ToList();

SelecetdCar.ItemsSource = items;

List<string> items2 = context.Buyers.Select(p => p.FullName).ToList();

SelecetdByer.ItemsSource = items2;

}

}

private void Button\_Click\_3(object sender, RoutedEventArgs e)

{

if (string.IsNullOrEmpty(cost.Text) || string.IsNullOrEmpty(SelecetdCar.Text) || string.IsNullOrEmpty(VinTextBox.Text) || string.IsNullOrEmpty(BrandTextBox.Text) || string.IsNullOrEmpty(year.Text) || string.IsNullOrEmpty(power.Text) || string.IsNullOrEmpty(pts.Text) || string.IsNullOrEmpty(color.Text) || string.IsNullOrEmpty(fio.Text) || string.IsNullOrEmpty(address.Text) || string.IsNullOrEmpty(city.Text) || string.IsNullOrEmpty(seria.Text) || string.IsNullOrEmpty(nomer.Text))

{

MessageBox.Show("Заполните все поля!");

}

else

{

var maker = new WordMaker("dkp.docs");

var items = new Dictionary<string, string>

{

{ "<DATE>", DateTime.Now.ToString("MM.dd.yyyy") },

{ "<MODEL>", SelecetdCar.Text },

{ "<VIN>", VinTextBox.Text },

{ "<BRAND>", BrandTextBox.Text },

{ "<YEAR>", year.Text },

{ "<NUMBEREN>", power.Text },

{ "<NUMBERFR>", pts.Text },

{ "<COLOR>", color.Text },

{ "<FIO2>", fio.Text },

{ "<ADRESS2>", address.Text },

{ "<CITY2>", city.Text },

{ "<SERIA2>", seria.Text },

{ "<NOMER2>", nomer.Text },

{ "<COST>", cost.Text },

};

maker.Process(items);

using (AutosallerContext context = new AutosallerContext())

{

var sale = new Sale

{

AutoId = int.Parse(HiddenCarId.Text),

BuyerId = int.Parse(HiddenByerId.Text),

Price = int.Parse(cost.Text),

WarrantyMonths = 12,

SaleDate = DateOnly.FromDateTime(DateTime.Parse(DateTime.Now.ToString("yyyy-MM-dd")))

};

context.Sales.Add(sale);

var car = context.Cars.FirstOrDefault(c => c.AutoId == int.Parse(HiddenCarId.Text));

if (car != null)

{

car.StatusId = 2;

context.Cars.Update(car);

}

context.SaveChanges();

}

MessageBox.Show("Документ сформирован и запись о продаже добавлена в базу данных.");

frame.Navigate(new Sales(frame));

}

}

private void SelecetdCar\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

if (SelecetdCar.SelectedItem != null)

{

string selectedModel = SelecetdCar.SelectedItem.ToString();

using (AutosallerContext context = new AutosallerContext())

{

context.Cars.ToList();

context.Brands.ToList();

var car = context.Cars.FirstOrDefault(c => c.Model == selectedModel);

if (car != null)

{

VinTextBox.Text = car.VinNumber;

BrandTextBox.Text = car.Brand.BrandName;

year.Text = car.YearOfManufacture.ToString();

power.Text = car.EnginePower;

pts.Text = car.PtsNumber;

color.Text = car.Color;

HiddenCarId.Text = car.AutoId.ToString();

}

}

}

}

private void SelecetdByer\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

if (SelecetdByer.SelectedItem != null)

{

string selectedModel = SelecetdByer.SelectedItem.ToString();

using (AutosallerContext context = new AutosallerContext())

{

context.Buyers.ToList();

context.Brands.ToList();

var byer = context.Buyers.FirstOrDefault(c => c.FullName == selectedModel);

if (byer != null)

{

fio.Text = byer.FullName;

city.Text = byer.CityAddress;

address.Text = byer.EmailAddress;

seria.Text = byer.SeriaPassport;

nomer.Text = byer.NumberPassport;

HiddenByerId.Text = byer.BuyerId.ToString();

}

}

}

}

}

}

Auth.xaml.cs

using AutoMir.Model;

using AutoMir.Service;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace AutoMir

{

/// <summary>

/// Логика взаимодействия для Auth.xaml

/// </summary>

public partial class Auth : Page

{

private Frame frame = null;

public static Employee user;

public Auth(Frame frame)

{

InitializeComponent();

this.frame = frame;

}

private void Auth\_Click(object sender, RoutedEventArgs e)

{

try

{

if (string.IsNullOrEmpty(login.Text) || string.IsNullOrEmpty(password.Text))

{

MessageBox.Show("Поля пустые");

}

else

{

Employee user2 = UserService.FindUserByLogin(login.Text);

if (user2 == null || user2.Password != password.Text)

{

MessageBox.Show("Неверный пароль или логин");

}

else

{

Auth.user = user2;

Security.user = user2;

MessageBox.Show("Вы вошли как - " + user.Post.PostName);

NavigationService.Navigate(new Cars(frame));

}

}

}

catch

{

MessageBox.Show("Error");

}

}

}

}

BuyerInfo.xaml.cs

using AutoMir.Model.Data;

using AutoMir.Model;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace AutoMir

{

/// <summary>

/// Логика взаимодействия для BuyerInfo.xaml

/// </summary>

public partial class BuyerInfo : Page

{

private Frame frame;

private int buyerID;

public BuyerInfo(int buyerID, Frame frame)

{

InitializeComponent();

this.buyerID = buyerID;

this.frame = frame;

using (AutosallerContext context = new AutosallerContext())

{

var buyer = context.Buyers.FirstOrDefault(b => b.BuyerId == buyerID);

if (buyer != null)

{

fio.Text = buyer.FullName;

phonenumber.Text = buyer.PhoneNumber;

city.Text = buyer.CityAddress;

email.Text = buyer.EmailAddress;

adress.Text = buyer.ResidentialAddress;

seria.Text = buyer.SeriaPassport;

nomer.Text = buyer.NumberPassport;

req.Text = buyer.CarBrandRequirements;

tech.Text = buyer.TechnicalRequirements;

bud.Text = buyer.Budget?.ToString();

}

}

}

private void ChangeBuyer(object sender, RoutedEventArgs e)

{

using (AutosallerContext context = new AutosallerContext())

{

var buyer = context.Buyers.FirstOrDefault(b => b.BuyerId == buyerID);

if (buyer != null)

{

buyer.FullName = fio.Text;

buyer.PhoneNumber = phonenumber.Text;

buyer.CityAddress = city.Text;

buyer.EmailAddress = email.Text;

buyer.ResidentialAddress = adress.Text;

buyer.SeriaPassport = seria.Text;

buyer.NumberPassport = nomer.Text;

buyer.CarBrandRequirements = req.Text;

buyer.TechnicalRequirements = tech.Text;

buyer.Budget = decimal.TryParse(bud.Text, out decimal budget) ? (decimal?)budget : null;

context.SaveChanges();

}

}

MessageBox.Show("Изменения сохранены");

}

}

}

Buyers.xaml.cs

using AutoMir.Model;

using AutoMir.Service;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace AutoMir

{

/// <summary>

/// Логика взаимодействия для Buyers.xaml

/// </summary>

public partial class Buyers : Page

{

private Frame frame;

private Buyer buyer;

public Buyers(Frame frame)

{

InitializeComponent();

this.frame = frame;

DataContext = buyer;

Employee user1 = Security.user;

EmpName.Text = user1.Post.PostName;

List<Buyer> buyers = BuyerService.GetBuyerList();

foreach (Buyer buyer in buyers)

{

ListProducts.Items.Add(buyer);

}

}

private void addBuy\_Click(object sender, RoutedEventArgs e)

{

frame.Navigate(new AddBuyer(frame));

}

private void search\_TextChanged(object sender, TextChangedEventArgs e)

{

List<Buyer> buyers = BuyerService.GetBuyerList();

if (!string.IsNullOrEmpty(search.Text))

{

buyers = buyers.Where(v => v.FullName.ToLower().Contains(search.Text)).ToList();

}

ListProducts.Items.Clear();

foreach (Buyer buyer in buyers)

{

ListProducts.Items.Add(buyer);

}

}

}

}

BuyerUserControl.xaml.cs

using AutoMir.Model;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace AutoMir

{

/// <summary>

/// Логика взаимодействия для BuyerUserControl.xaml

/// </summary>

public partial class BuyerUserControl : UserControl

{

public BuyerUserControl()

{

InitializeComponent();

}

private Frame FindParentFrame()

{

DependencyObject parent = this;

while (parent != null)

{

if (parent is Frame frame)

{

return frame;

}

parent = VisualTreeHelper.GetParent(parent);

}

return null;

}

private void Grid\_MouseDown(object sender, MouseButtonEventArgs e)

{

if (this.DataContext is Buyer buyer)

{

var buyerID = buyer.BuyerId;

var parentFrame = FindParentFrame();

if (parentFrame != null)

{

parentFrame.Navigate(new BuyerInfo(buyerID, parentFrame));

}

else

{

MessageBox.Show("Error: Parent frame not found.");

}

}

else

{

MessageBox.Show("Error: DataContext is not set or is not of type Automobile.");

}

}

}

}

CarInfo.xaml.cs

using AutoMir.Model;

using AutoMir.Model.Data;

using AutoMir.Service;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace AutoMir

{

/// <summary>

/// Логика взаимодействия для CarInfo.xaml

/// </summary>

public partial class CarInfo : Page

{

private Frame frame;

private int carId;

public CarInfo(int carId, Frame frame)

{

InitializeComponent();

this.frame = frame;

this.carId = carId;

CarDataLoad();

}

private void CarDataLoad()

{

using (AutosallerContext context = new AutosallerContext())

{

var car = context.Cars.Find(carId);

if (car == null)

{

MessageBox.Show("Car not found");

return;

}

List<string> brandItems = context.Brands.Select(p => p.BrandName).ToList();

BrandIdTextBox.ItemsSource = brandItems;

BrandIdTextBox.SelectedItem = context.Brands.FirstOrDefault(b => b.BrandId == car.BrandId)?.BrandName;

List<string> transmissionItems = context.Transmissions.Select(p => p.TransmissionType).ToList();

TransmissionIdTextBox.ItemsSource = transmissionItems;

TransmissionIdTextBox.SelectedItem = context.Transmissions.FirstOrDefault(t => t.TransmissionId == car.TransmissionId)?.TransmissionType;

List<string> bodyItems = context.Bodies.Select(p => p.BodyType).ToList();

BodyIdTextBox.ItemsSource = bodyItems;

BodyIdTextBox.SelectedItem = context.Bodies.FirstOrDefault(b => b.BodyId == car.BodyId)?.BodyType;

List<string> conditionItems = context.Conditionautos.Select(p => p.ConditionName).ToList();

ConditionIdTextBox.ItemsSource = conditionItems;

ConditionIdTextBox.SelectedItem = context.Conditionautos.FirstOrDefault(c => c.ConditionId == car.ConditionId)?.ConditionName;

List<string> statusItems = context.Statusautos.Select(p => p.StatusName).ToList();

StatusIdTextBox.ItemsSource = statusItems;

StatusIdTextBox.SelectedItem = context.Statusautos.FirstOrDefault(s => s.StatusId == car.StatusId)?.StatusName;

ModelTextBox.Text = car.Model;

CountryOfProductionTextBox.Text = car.CountryOfProduction;

VinNumberTextBox.Text = car.VinNumber;

PtsNumberTextBox.Text = car.PtsNumber;

YearOfManufactureTextBox.Text = car.YearOfManufacture?.ToString();

EnginePowerTextBox.Text = car.EnginePower;

EquipmentTextBox.Text = car.Equipment;

ColorTextBox.Text = car.Color;

ProbegTextBox.Text = car.Probeg?.ToString();

NumberOfOwnersTextBox.Text = car.NumberOfOwners?.ToString();

ImageTextBox.Text = car.Image;

image.Source = new BitmapImage(new Uri(car.Image));

AskingPriceTextBox.Text = car.AskingPrice?.ToString();

}

}

private void SaveChangesButton\_Click(object sender, RoutedEventArgs e)

{

using (AutosallerContext context = new AutosallerContext())

{

var car = context.Cars.Find(carId);

if (car == null)

{

MessageBox.Show("Car not found");

return;

}

car.BrandId = context.Brands.FirstOrDefault(b => b.BrandName == BrandIdTextBox.SelectedItem.ToString())?.BrandId;

car.Model = ModelTextBox.Text;

car.CountryOfProduction = CountryOfProductionTextBox.Text;

car.VinNumber = VinNumberTextBox.Text;

car.PtsNumber = PtsNumberTextBox.Text;

car.YearOfManufacture = int.TryParse(YearOfManufactureTextBox.Text, out int yearOfManufacture) ? (int?)yearOfManufacture : null;

car.EnginePower = EnginePowerTextBox.Text;

car.TransmissionId = context.Transmissions.FirstOrDefault(t => t.TransmissionType == TransmissionIdTextBox.SelectedItem.ToString())?.TransmissionId;

car.Equipment = EquipmentTextBox.Text;

car.Color = ColorTextBox.Text;

car.BodyId = context.Bodies.FirstOrDefault(b => b.BodyType == BodyIdTextBox.SelectedItem.ToString())?.BodyId;

car.ConditionId = context.Conditionautos.FirstOrDefault(c => c.ConditionName == ConditionIdTextBox.SelectedItem.ToString())?.ConditionId;

car.StatusId = context.Statusautos.FirstOrDefault(s => s.StatusName == StatusIdTextBox.SelectedItem.ToString())?.StatusId;

car.Probeg = int.TryParse(ProbegTextBox.Text, out int probeg) ? (int?)probeg : null;

car.NumberOfOwners = int.TryParse(NumberOfOwnersTextBox.Text, out int numberOfOwners) ? (int?)numberOfOwners : null;

car.Image = ImageTextBox.Text;

car.AskingPrice = decimal.TryParse(AskingPriceTextBox.Text, out decimal askingPrice) ? (decimal?)askingPrice : null;

context.SaveChanges();

}

MessageBox.Show("Изменения внесены");

frame.Navigate(new Cars(frame));

}

}

}

Cars.xaml.cs

using AutoMir.Model;

using AutoMir.Model.Data;

using AutoMir.Service;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace AutoMir

{

/// <summary>

/// Логика взаимодействия для Cars.xaml

/// </summary>

public partial class Cars : Page

{

private Frame frame;

private Car automobile1;

public Cars(Frame frame)

{

InitializeComponent();

this.frame = frame;

Employee user1 = Security.user;

DataContext = automobile1;

EmpName.Text = user1.Post.PostName;

List<Car> Automobiles = CarListService.GetAutoList();

foreach (Car automobile in Automobiles)

{

ListProducts.Items.Add(automobile);

}

}

private void addcar\_Click(object sender, RoutedEventArgs e)

{

frame.Navigate(new AddCar(frame));

}

private void search\_TextChanged(object sender, TextChangedEventArgs e)

{

List<Car> Automobiles = CarListService.GetAutoList();

if (!string.IsNullOrEmpty(search.Text))

{

Automobiles = Automobiles.Where(v => v.Model.ToLower().Contains(search.Text) || v.Brand.BrandName.ToLower().Contains(search.Text)).ToList();

}

ListProducts.Items.Clear();

foreach (Car automobile in Automobiles)

{

ListProducts.Items.Add(automobile);

}

}

private void brandSel\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

List<Car> Automobiles = CarListService.GetAutoList();

switch (tranSel.SelectedIndex)

{

case 1:

Automobiles = Automobiles.Where(v => v.Transmission.TransmissionId == 1).ToList();

break;

case 2:

Automobiles = Automobiles.Where(v => v.Transmission.TransmissionId == 2).ToList();

break;

}

ListProducts.Items.Clear();

foreach (Car automobile in Automobiles)

{

ListProducts.Items.Add(automobile);

}

}

private void brandSel\_SelectionChanged\_1(object sender, SelectionChangedEventArgs e)

{

List<Car> Automobiles = CarListService.GetAutoList();

switch (brandSel.SelectedIndex)

{

case 1:

Automobiles = Automobiles.Where(v => v.Brand.BrandId == 1).ToList();

break;

case 2:

Automobiles = Automobiles.Where(v => v.Brand.BrandId == 2).ToList();

break;

case 3:

Automobiles = Automobiles.Where(v => v.Brand.BrandId == 3).ToList();

break;

case 4:

Automobiles = Automobiles.Where(v => v.Brand.BrandId == 4).ToList();

break;

case 5:

Automobiles = Automobiles.Where(v => v.Brand.BrandId == 5).ToList();

break;

case 6:

Automobiles = Automobiles.Where(v => v.Brand.BrandId == 6).ToList();

break;

case 7:

Automobiles = Automobiles.Where(v => v.Brand.BrandId == 7).ToList();

break;

case 8:

Automobiles = Automobiles.Where(v => v.Brand.BrandId == 8).ToList();

break;

case 9:

Automobiles = Automobiles.Where(v => v.Brand.BrandId == 9).ToList();

break;

}

ListProducts.Items.Clear();

foreach (Car automobile in Automobiles)

{

ListProducts.Items.Add(automobile);

}

}

private void kuzSel\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

List<Car> Automobiles = CarListService.GetAutoList();

switch (kuzSel.SelectedIndex)

{

case 1:

Automobiles = Automobiles.Where(v => v.Body.BodyId == 1).ToList();

break;

case 2:

Automobiles = Automobiles.Where(v => v.Body.BodyId == 2).ToList();

break;

case 3:

Automobiles = Automobiles.Where(v => v.Body.BodyId == 3).ToList();

break;

case 4:

Automobiles = Automobiles.Where(v => v.Body.BodyId == 4).ToList();

break;

}

ListProducts.Items.Clear();

foreach (Car automobile in Automobiles)

{

ListProducts.Items.Add(automobile);

}

}

private void condSel\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

List<Car> Automobiles = CarListService.GetAutoList();

switch (condSel.SelectedIndex)

{

case 1:

Automobiles = Automobiles.Where(v => v.Condition.ConditionId == 1).ToList();

break;

case 2:

Automobiles = Automobiles.Where(v => v.Condition.ConditionId == 2).ToList();

break;

}

ListProducts.Items.Clear();

foreach (Car automobile in Automobiles)

{

ListProducts.Items.Add(automobile);

}

}

}

}

CarUserControl.xaml.cs

using AutoMir.Model;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace AutoMir

{

/// <summary>

/// Логика взаимодействия для CarUserControl.xaml

/// </summary>

public partial class CarUserControl : UserControl

{

public CarUserControl()

{

InitializeComponent();

}

private void Grid\_MouseDown(object sender, MouseButtonEventArgs e)

{

if (this.DataContext is Car automobile)

{

var autoId = automobile.AutoId;

var parentFrame = FindParentFrame();

if (parentFrame != null)

{

parentFrame.Navigate(new CarInfo(autoId, parentFrame));

}

else

{

MessageBox.Show("Error: Parent frame not found.");

}

}

else

{

MessageBox.Show("Error: DataContext is not set or is not of type Automobile.");

}

}

private Frame FindParentFrame()

{

DependencyObject parent = this;

while (parent != null)

{

if (parent is Frame frame)

{

return frame;

}

parent = VisualTreeHelper.GetParent(parent);

}

return null;

}

}

}

Sales.xaml.cs

using AutoMir.Model;

using AutoMir.Service;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace AutoMir

{

/// <summary>

/// Логика взаимодействия для Sales.xaml

/// </summary>

public partial class Sales : Page

{

private Frame frame;

private Sale sales;

public Sales(Frame frame)

{

InitializeComponent();

this.frame = frame;

DataContext = sales;

Employee user1 = Security.user;

EmpName.Text = user1.Post.PostName;

List<Sale> sale = SalesService.GetSaleList();

foreach (Sale sales in sale)

{

ListProducts.Items.Add(sales);

}

}

private void CreateSale\_Click(object sender, RoutedEventArgs e)

{

frame.Navigate(new AddSale(frame));

}

private void search\_TextChanged(object sender, TextChangedEventArgs e)

{

List<Sale> sale = SalesService.GetSaleList();

if (!string.IsNullOrEmpty(search.Text))

{

sale = sale.Where(v => v.Buyer.FullName.ToLower().Contains(search.Text)).ToList();

}

ListProducts.Items.Clear();

foreach (Sale sales in sale)

{

ListProducts.Items.Add(sales);

}

}

private void datesale\_MouseDown(object sender, MouseButtonEventArgs e)

{

}

}

}

Класс WordMaker.cs

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using Word = Microsoft.Office.Interop.Word;

namespace AutoMir

{

internal class WordMaker

{

private FileInfo \_fileInfo;

public WordMaker(string filename)

{

if (File.Exists(filename))

{

\_fileInfo = new FileInfo(filename);

}

else

{

throw new ArgumentException("Файл не сущестсвует");

}

}

internal bool Process(Dictionary<string, string> items)

{

try

{

var app = new Word.Application();

Object file = \_fileInfo.FullName;

Object missing = Type.Missing;

app.Documents.Open(file);

foreach (var item in items)

{

Word.Find find = app.Selection.Find;

find.Text = item.Key;

find.Replacement.Text = item.Value;

Object wrap = Word.WdFindWrap.wdFindContinue;

Object replace = Word.WdReplace.wdReplaceAll;

find.Execute(FindText: Type.Missing,

MatchCase: false,

MatchWholeWord: false,

MatchWildcards: false,

MatchSoundsLike: missing,

MatchAllWordForms: false,

Forward: true,

Wrap: wrap,

Format: false,

ReplaceWith: missing, Replace: replace);

}

Object newFileName = Path.Combine(\_fileInfo.DirectoryName, DateTime.Now.ToString("yyyyMMdd HHmmss") + \_fileInfo.Name);

app.ActiveDocument.SaveAs2(newFileName);

app.ActiveDocument.Close();

app.Quit();

return true;

}

catch (Exception ex) { MessageBox.Show("Error"); }

return false;

}

}

}

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. ГОСТ 19.701-90 ЕСПД. Схемы алгоритмов, программ, данных и систем. Обозначения условные и правила выполнения [Текст]. – Введ. 1992–01–01. – М.: Стандартинформ, 2010 – 24 с.
2. ГОСТ 34.601-90 Информационная технология. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Стадии создания [Текст]. – Введ. 1990-01-01. – М.: Стандартинформ, 2007.
3. ГОСТ Р 51583-2014. Защита информации. Порядок создания автоматизированных систем в защищенном исполнении. Общие положения [Текст]. – Введ. 2014-09-01. – М.: Стандартинформ, 2018 – 20 с.
4. C# и .NET: разработка приложений с использованием WPF и MySQL : учебное пособие. – Санкт-Петербург : Питер, 2025. – 368 с. : ил. – (Серия «Библиотека программиста»). – ISBN 978-5-4461-4131-9. – Текст : непосредственный.
5. Гордеев, С. И. Организация баз данных в 2 ч. Часть 1: учебник для среднего профессионального образования / С. И. Гордеев, В. Н. Волошина. — 2-е изд., испр. и доп. — Москва : Издательство Юрайт, 2025. — 310 с. — (Профессиональное образование). — ISBN 978-5-534-11626-7. — Текст: электронный // Образовательная платформа Юрайт [сайт]. — URL: <https://urait.ru/bcode/518510> (дата обращения: 28.01.2025).
6. Гордеев, С. И. Организация баз данных в 2 ч. Часть 2 : учебник для вузов / С. И. Гордеев, В. Н. Волошина. — 2-е изд., испр. и доп. — Москва: Издательство Юрайт, 2025. — 513 с. — (Высшее образование). — ISBN 978-5-534-04470-6. — Текст: электронный // Образовательная платформа Юрайт [сайт]. — URL: <https://urait.ru/bcode/566521> (дата обращения: 19.02.2025).
7. Грокаем C# и WPF / Р. Хайнлайн. – Москва : Издательство Питер, 2023. – 304 с. – (Серия «Библиотека программиста»). – ISBN 978-5-4461-1234-0. – Текст : непосредственный.
8. Основы баз данных с использованием MySQL / С. А. Смирнов. – Москва : Издательство Юрайт, 2024. – 320 с. – (Серия «Профессиональное образование»). – ISBN 978-5-534-12345-6. – Текст : непосредственный.
9. Основы DevOps: принципы и практика для C# разработчиков / С. А. Смирнов, В. Н. Волошина. – Москва : Издательство Юрайт, 2025. – 350 с. – (Серия «Профессиональное образование»). – ISBN 978-5-534-12347-0. – Текст : непосредственный.
10. Паттерны проектирования для C# и WPF / Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес. – Москва : Издательство Вильямс, 2020. – 368 с. – ISBN 978-5-8459-2100-7. – Текст : непосредственный.
11. Программирование на C#: от основ до создания приложений с WPF / М. Н. Кузнецов. – Санкт-Петербург : БХВ-Петербург, 2025. – 512 с. – ISBN 978-5-9775-4123-4. – Текст : непосредственный.
12. Современные технологии разработки на C# и WPF / А. В. Иванов, Б. С. Петров, В. Г. Сидоров [и др.]. – Москва : Издательство Лань, 2024. – 420 с. – ISBN 978-5-8114-5678-9. – Текст : непосредственный.
13. Тестирование программного обеспечения на C#: основы и практика / А. В. Петров, Б. С. Иванов. – Москва : Издательство Лань, 2023. – 280 с. – ISBN 978-5-8114-5679-6. – Текст : непосредственный.
14. Управление проектами в IT с использованием C# и WPF / А. Б. Шестаков, В. В. Иванов. – Москва : Издательство Инфра-М, 2024. – 320 с. – ISBN 978-5-16-015679-8. – Текст : непосредственный.
15. Чистая архитектура для C# разработчиков / Роберт Мартин. – Москва : Издательство Вильямс, 2022. – 352 с. – ISBN 978-5-8459-2210-3. – Текст : непосредственный.
16. MySQL документация : [официальный сайт]. – URL: <https://dev.mysql.com/doc/> (дата обращения: 20.03.2025) – Текст : электронный.
17. MySQL для разработчиков: практическое руководство / Е. П. Моргунов. – Москва : Postgres Professional, 2024. – 400 с. – ISBN 978-5-9909445-3-6. – Текст : непосредственный.
18. C# – к вершинам мастерства: Лаконичное и эффективное программирование / Лусиану Рамальо ; пер. с англ. А. А. Слинкина. – 2-е изд. – Москва : МК Пресс, 2022. – 898 с. : ил. – ISBN 978-5-9909445-2-9. – Текст : непосредственный.
19. WPF и MySQL: современная разработка приложений / А. В. Иванов. – Москва : Издательство Лань, 2024. – 300 с. – ISBN 978-5-8114-5677-2. – Текст : непосредственный.
20. Разработка приложений с использованием WPF и MySQL / Б. С. Петров. – Москва : Издательство Инфра-М, 2024. – 350 с. – ISBN 978-5-16-015680-4. – Текст : непосредственный.