Оглавление

Введение

Объектом исследования курсовой работы является взаимодействие работников автосалона с клиентами и поставщиками. В создаваемой базе данных будет храниться информация обо всей работе автосалона. Для удобства использования этой информации сотрудникам необходима автоматизированная система, позволяющая добавлять, изменять и удалять информацию.

Целью данного курсового проекта является создание БД, хранящей всю информацию о деятельности автосалона, а также удобного пользовательского интерфейса для взаимодействия с БД.

Основываясь на вышеупомянутой информации, можно определить задачи данного курсового проекта. Первая задача это проектирование самой БД. Для этого необходимо изучить предметную область, чтобы оценить функционал приложений предложенных для решения задачи управления автосалоном. Далее нужно провести концептуальное проектирование, в ходе которого сформируется формализованное описание предметной области нашей БД. Затем необходимо заняться логическим проектированием, чтобы определить состав и структуру таблиц базы данных на основе результатов концептуального проектирования. Последним пунктом будет физическое проектирование, в ходе которого будет создана схема БД нашего курсового проекта. Второй основной задачей данного курсового проекта является создание пользовательского интерфейса для взаимодействия сотрудников с ранее созданной БД.

ГЛАВА 1. Проектирование структуры базы данных

1.1. Анализ предметной области

Предметную область можно определить как сферу человеческой деятельности, выделенную и описанную согласно установленным критериям.В описываемое понятие должны входить сведения об ее элементах, явлениях, отношениях и процессах, отражающих различные аспекты этой деятельности.

Одна из первых задач, с решением которых сталкивается разработчик программной системы – это изучение, осмысление и анализ предметной области. Дело в том, что предметная область сильно влияет на все аспекты проекта: требования к системе, взаимодействие с пользователем, модель хранения данных, реализацию и т.д.

Анализ предметной области позволяет выделить ее сущности, определить первоначальные требования к функциональности и определить границы проекта.

В данной работе в качестве предметной области рассматривается деятельность автосалона.

Актуальность выполнения данной работы заключается в сохраняющемся и растущим по сей день спросе на автомобили как владельцев бизнесов, так и отдельных людей. Раньше продажи проходили только в автосалоне и получить актуальные данные о цене и характеристиках автомобиля можно было непосредственно при посещении автосалона. Сейчас, при вхождении человечества в цифровую эру, ведение такого бизнеса становится намного проще, за счет множества готовых продуктов, включающих в себя инструменты для хранения, передачи и анализа информации об автосалоне

Основными задачами, решаемыми в данной предметной области, являются:

- учет покупателей;
- заказ у поставщиков;
- предпродажная подготовка;

- учет комплектаций автомобилей;
- учет транспортных средств, включая их характеристики;
- поиск и подбор автомобилей по произвольным критериям;
- учет событий: покупка автомобиля, продажа автомобиля, тест-драйв и т.д.;
- оформление договоров купли-продажи;
- анализ продаж автомобилей определенных марок.

Существует множество программных продуктов, решающих задачи в данной области:

− сайты продажи поддержанных автомобилей (Auto.ru, Drom.ru, Avito.ru, rolf-probeg.ru и другие);

− программный продукт «АвтоДилер» от компании АвтоДилер [3];

− программный продукт 1С Альфа-Авто и другие.

Сервисы по продажи автомобилей предназначены для автоматизации и оптимизации всего процесса продажи автомобиля, начиная с поиска групп потенциальных покупателей и заканчивая заключением сделок на независимой платформе.

Большинство продуктов в этой категории позволяют автоматизировать размещение объявлений на собственных веб-сайтах. Для хранения информации об автомобилях в продаже может использоваться как собственная база данных, так и несколько независимых. Этот тип программного обеспечения, по существу, устраняет всю бумажную работу, связанную с продажей автомобилей.

В качестве первого программного продукта рассмотрим сайт по продаже автомобилей «Auto.ru» (https://auto.ru/moskva/) [1]. Auto.ru – это один из крупнейиших сервивов по продаже автомобилей в России, Украине и странах СНГ. Информационная платформа предназначена для организации покупки и продажи транспортных средств как автодилерами так и собственниками авто.

Портал для покупки и продажи автомобилей Auto.ru предоставляет

следующие основные возможности:

- Управление автопарком. Позволяет создавать, редактировать, публиковать, удалять и обновлять объявления (рис.1.1), обеспечивая доступ к объявлению всем заинтересованным покупателям открыто или по прямой ссылке. Сервис позволяет опубликовать весь объем информации об автомобиле, начиная с объема двигателя и заканчивая участием в ДТП и повреждениями как косметическими так и конструктивными (рис.1.2).

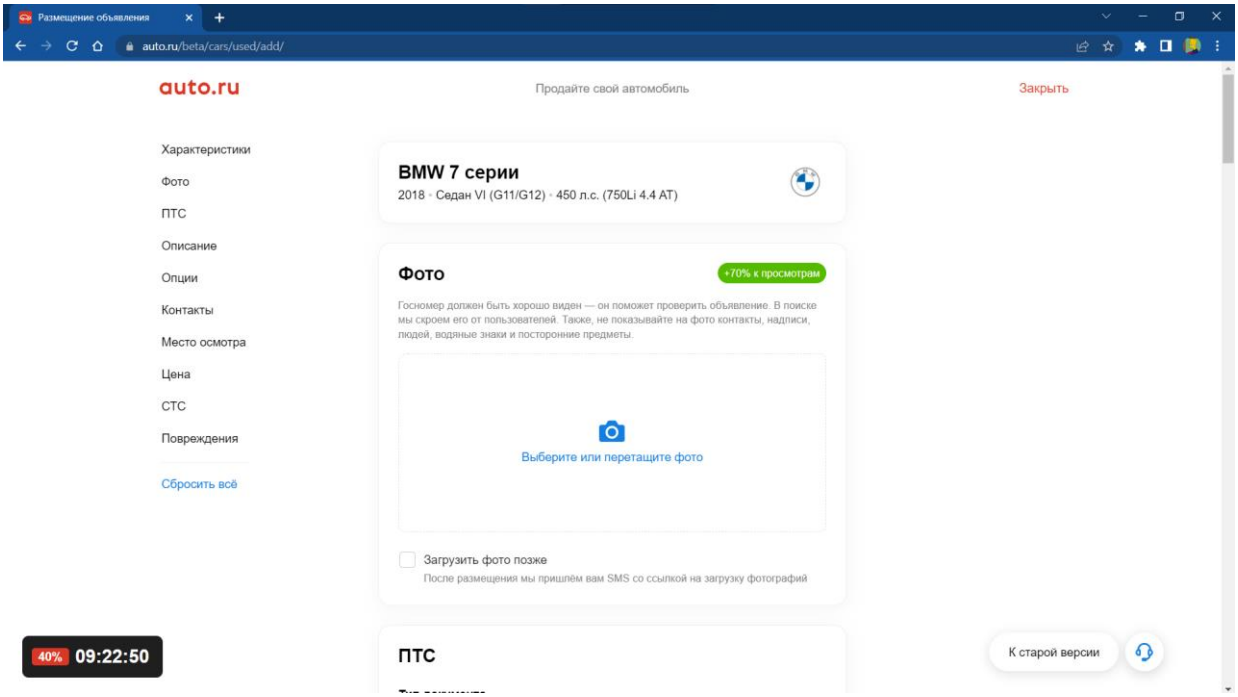


Рис 1.1 Размещение объявления

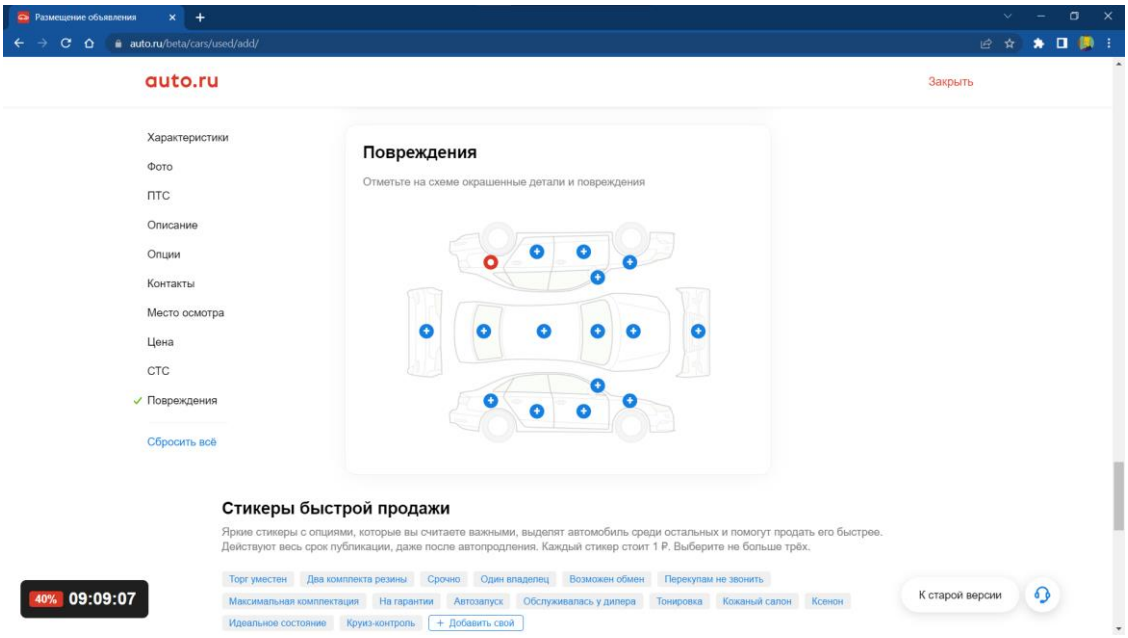


Рис 1.2 Настройка информации о повреждениях

- Поиск объявления. Возможность найти интересующий автомобиль в нужной комплектации с удовлетворяющей ценой (рис.1.3).

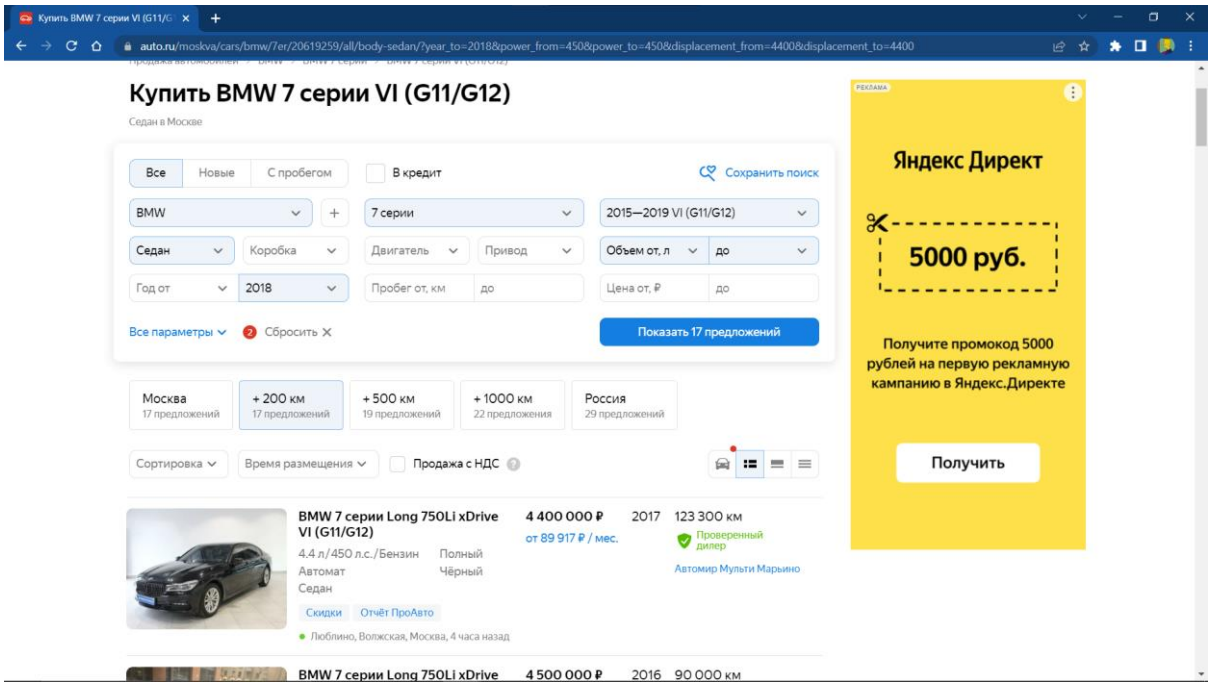


Рис 1.3 Поиск интересующего автомобиля

- Сервис позволяет гибко сортировать и фильтровать объявления по множеству критериев (рис.1.4).

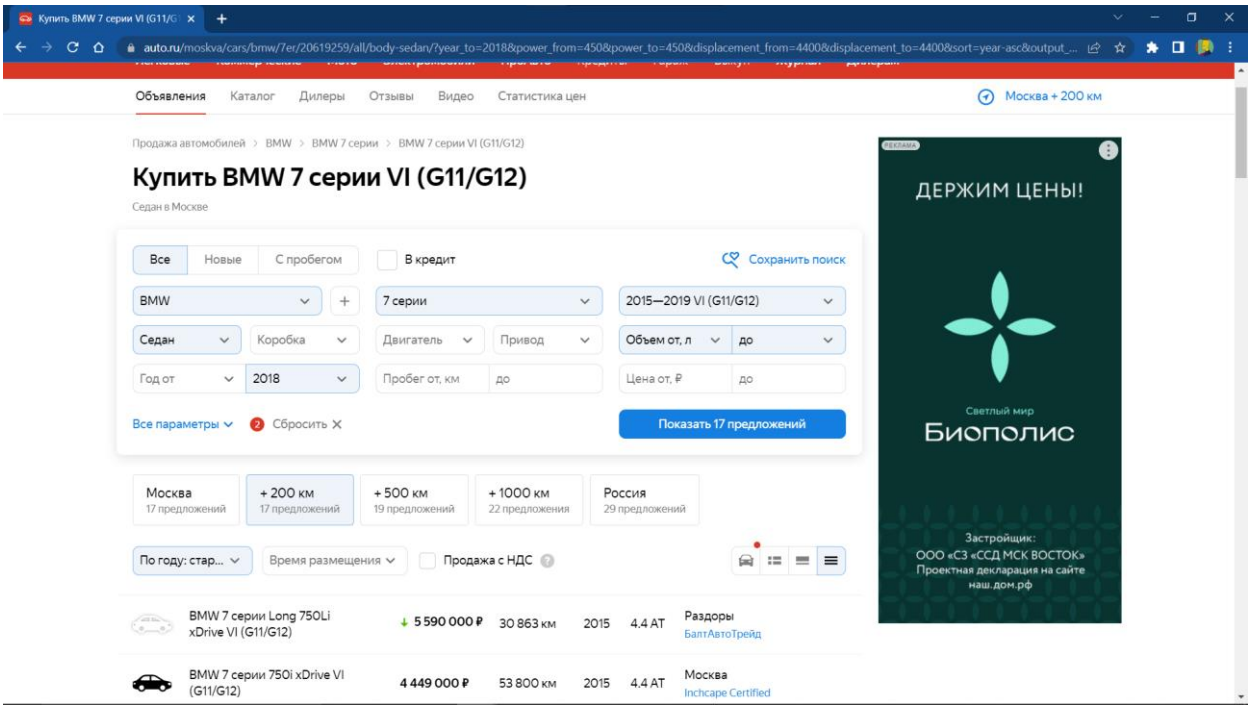


Рис 1.4 Сортировка объявлений по дате публикации на сайте

- Уведомления об объявлениях. Сервис позволяет настроить

автоматическую рассылку уведомлений при появлении в продаже интересующего автомобиля либо при отклике покупателя на выложенное объявление (рис.1.5).

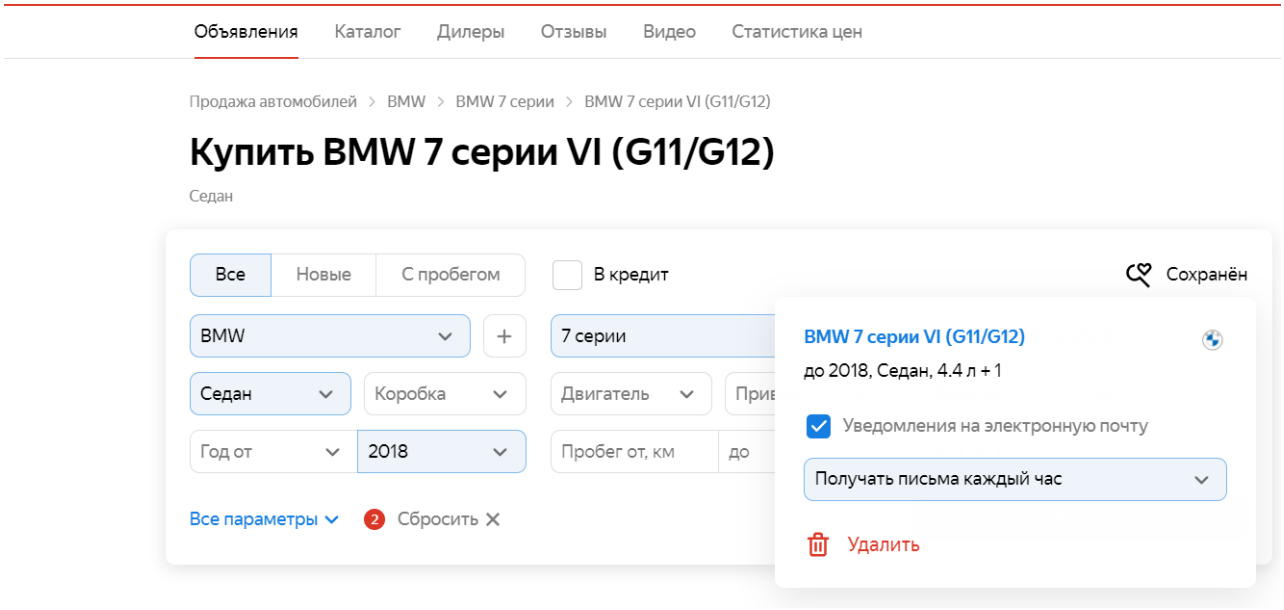


Рис 1.5 Настройка уведомлений о новых объявлениях по указанному автомобилю

В качестве второго программного продукта рассмотрим 1С:Альфа-Авто (https://1c-alfa-avto.ru/) [2]. Программный продукт **1С Альфа-Авто** — программа для автосервиса (СТО), учета запчастей и автосалонов. Среди основных функций данной программы стоит выделить:

- Обработка всех типов входящих обращений – телефонные звонки, визиты, электронные письма. В программе предусмотрен специализированный интерфейс для сотрудника ресепшена и специалиста колл-центра. В программе фиксируются входящие контакты с клиентом – звонки, визиты, письма – и полученные данные передаются продавцам автомобилей, продавцам запчастей, сервисным консультантам (рис.1.6).

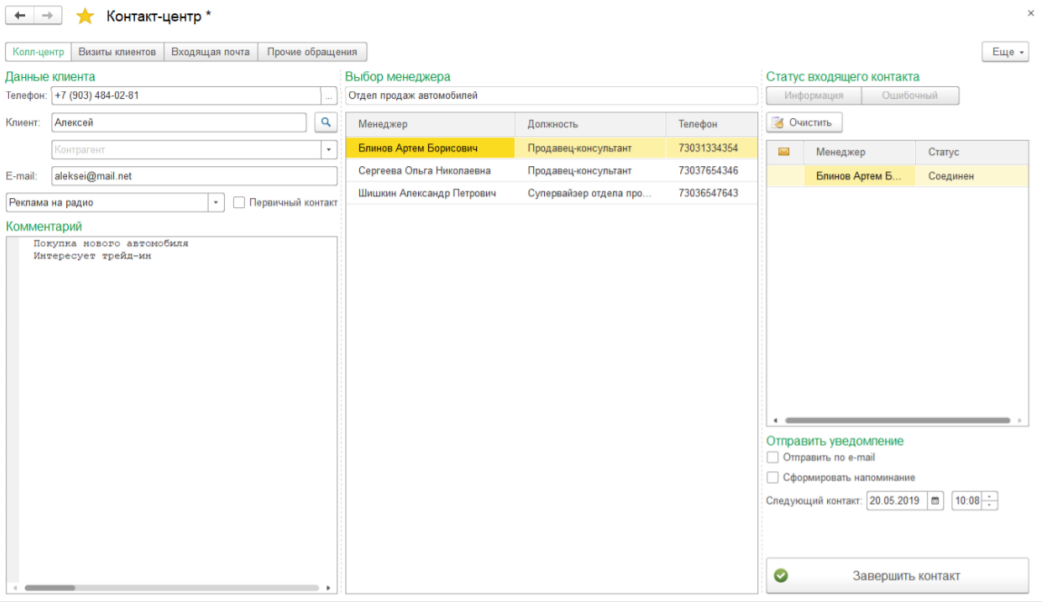


Рис 1.6 Интерфейс для сотрудника ресепшена и специалиста колл-центра

- Ведение сделок с клиентами по рабочим листам. Рабочий лист предназначен для проведения предпродажной работы с клиентом. В рабочем листе сохраняется информация о клиенте и основные параметры сделки (рис.1.7).

- Удобное конфигурирование автомобиля. Специальная форма для поиска автомобиля на складе компании или в размещенных у производителя заказах по различным параметрам модели: цвету, типу КПП и т.д. (рис.1.8).

- Тест-драйв – планирование и проведение. В системе предусмотрено ведение парка автомобилей тест-драйва, планирование тест-драйва и результатов тест-драйва автомобиля. Факт тест-драйва отражается соответствующим документом. Внутри документа «Тест-драйв» содержатся данные клиента, маршрут поездки, автомобиль тест-драйва, время начала и окончания, пробег автомобиля на начало и конец поездки (рис.1.9).

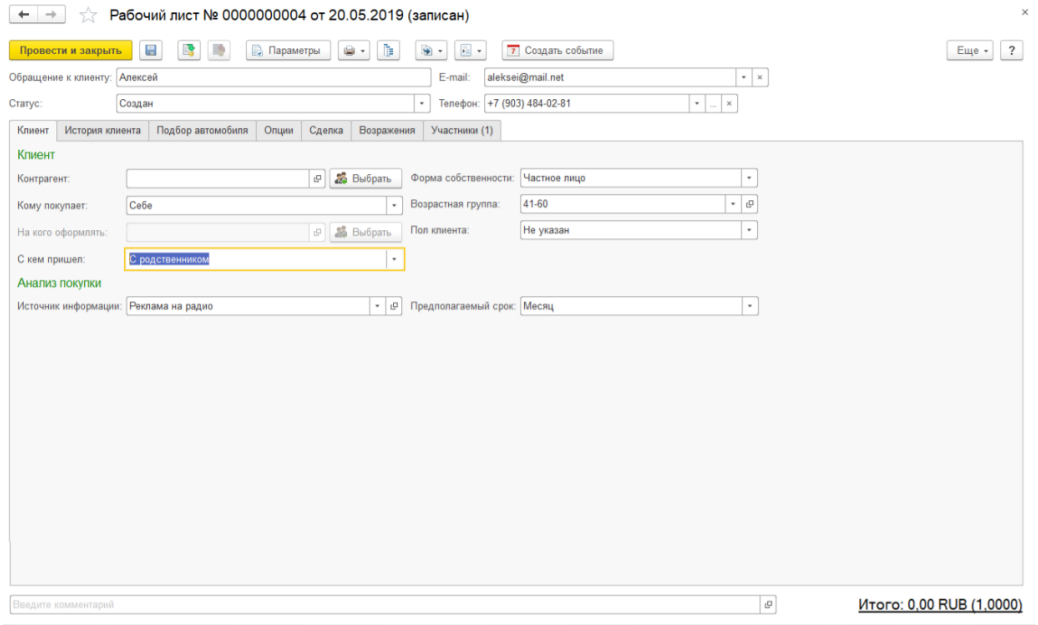


Рис 1.7 Рабочий лист

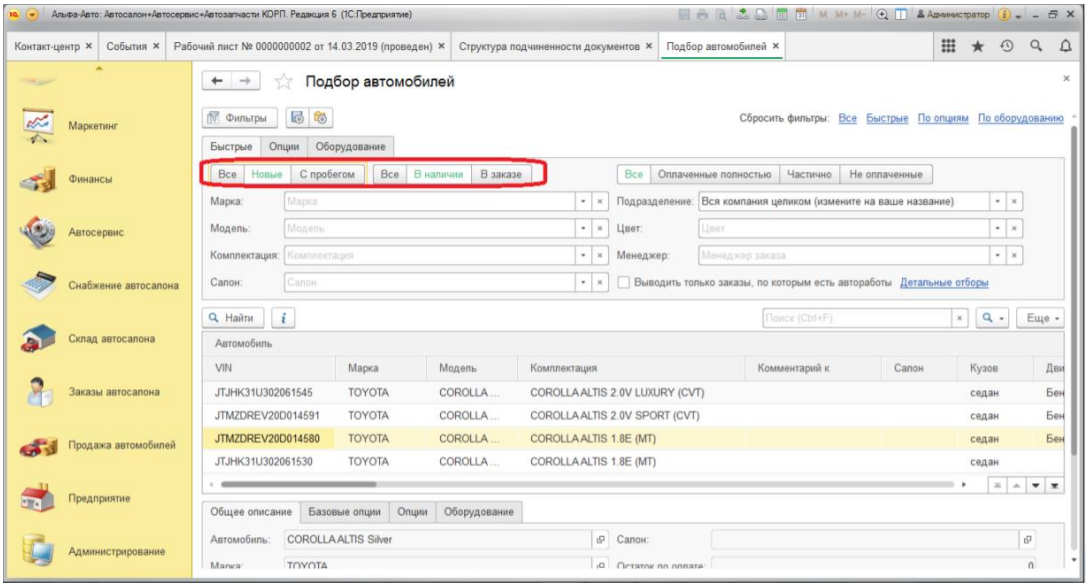


Рис 1.8 Конфигуратор автомобиля

- Подбор и резервирование автомобиля из наличия на складе.
- Заказ клиента на автомобиль. Для перехода от предпродажной деятельности к заключению предварительного договора купли-продажи предназначен документ «Заказ клиента на автомобиль». В документе указывается окончательная конфигурация закупаемого автомобиля (рис.1.10).

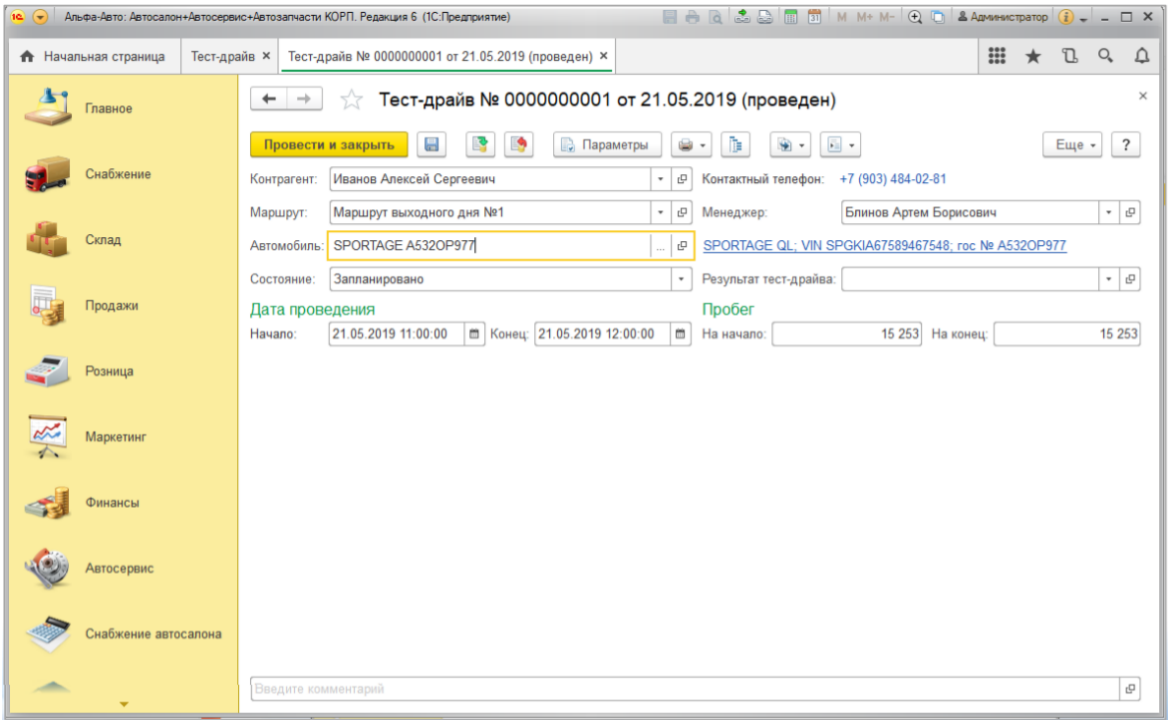


Рис 1.9 Планирование тест-драйва

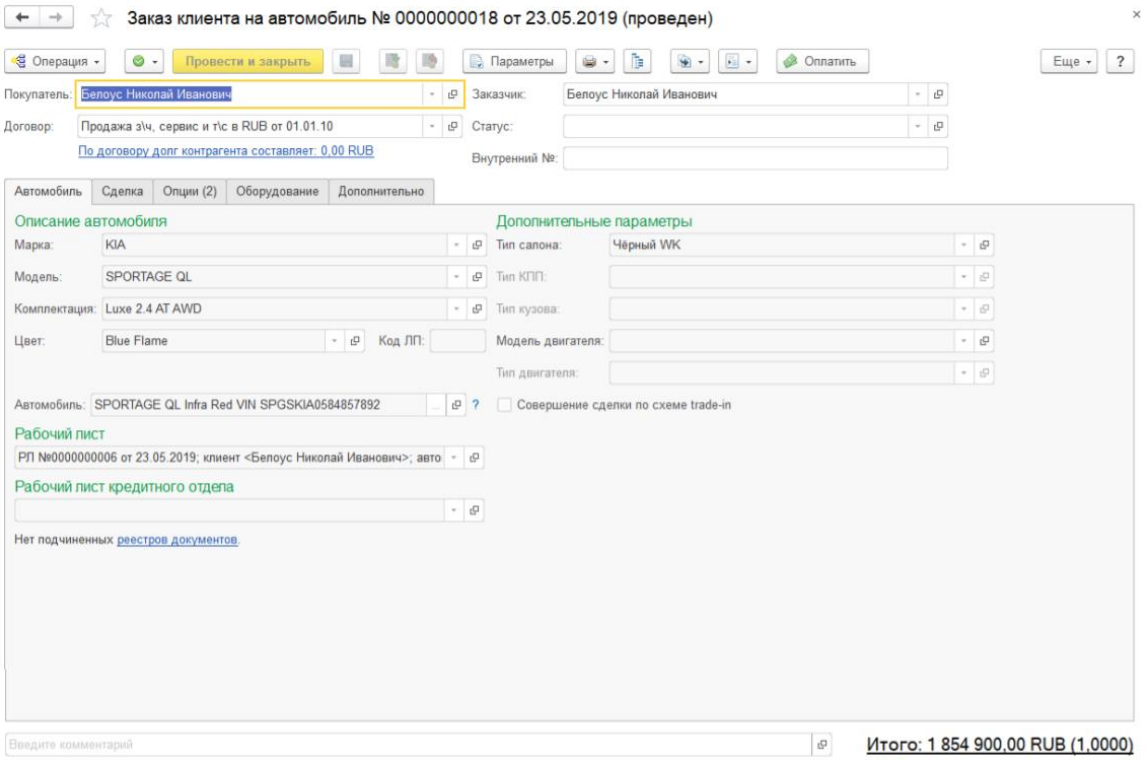


Рис 1.10 Заказ клиента на автомобиль

- Анализ рабочих листов. Для углубленного анализа рабочих листов предназначена аналитическая часть системы с расширенным списком отчётов (рис.1.11).

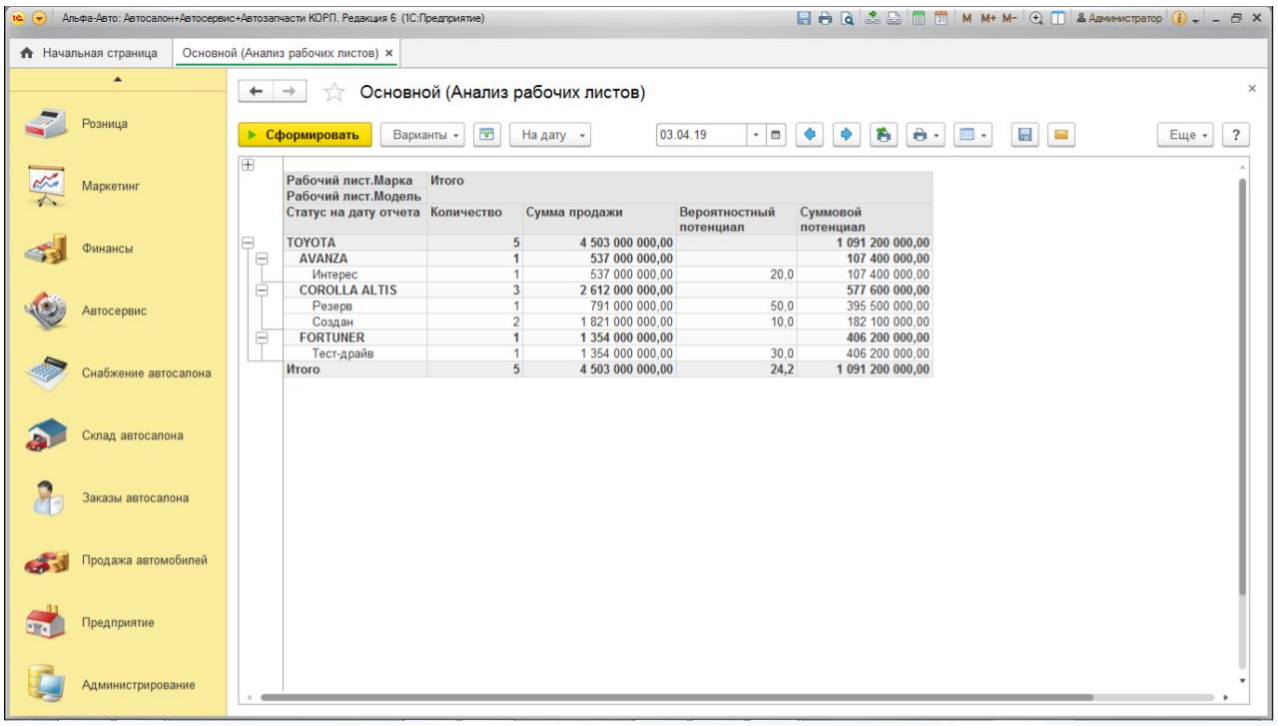


Рис 1.11 Анализ рабочих листов

- Выдача автомобиля. Факт отгрузки автомобиля клиенту отражается при помощи документа «Реализация автомобилей». Одной реализацией можно отгрузить список автомобилей. При этом производятся проверки по взаиморасчетам с клиентом, доступности автомобиля и его комплектации согласно заказу на автомобиль (рис.1.12).

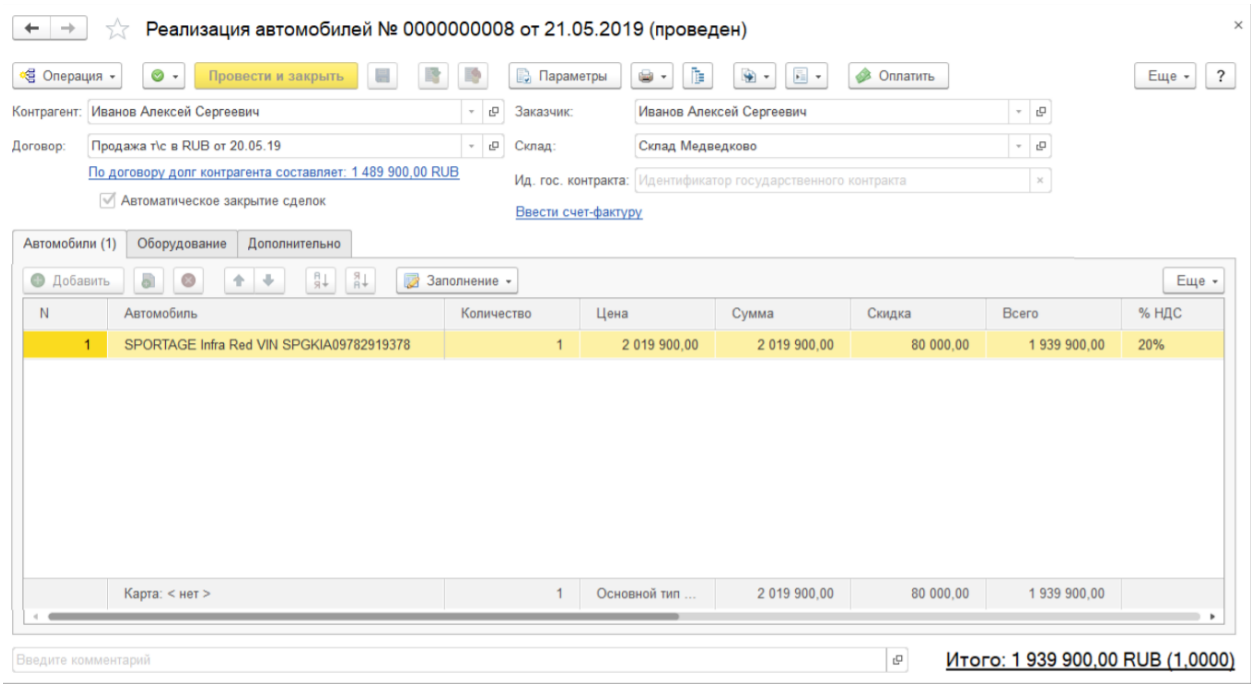
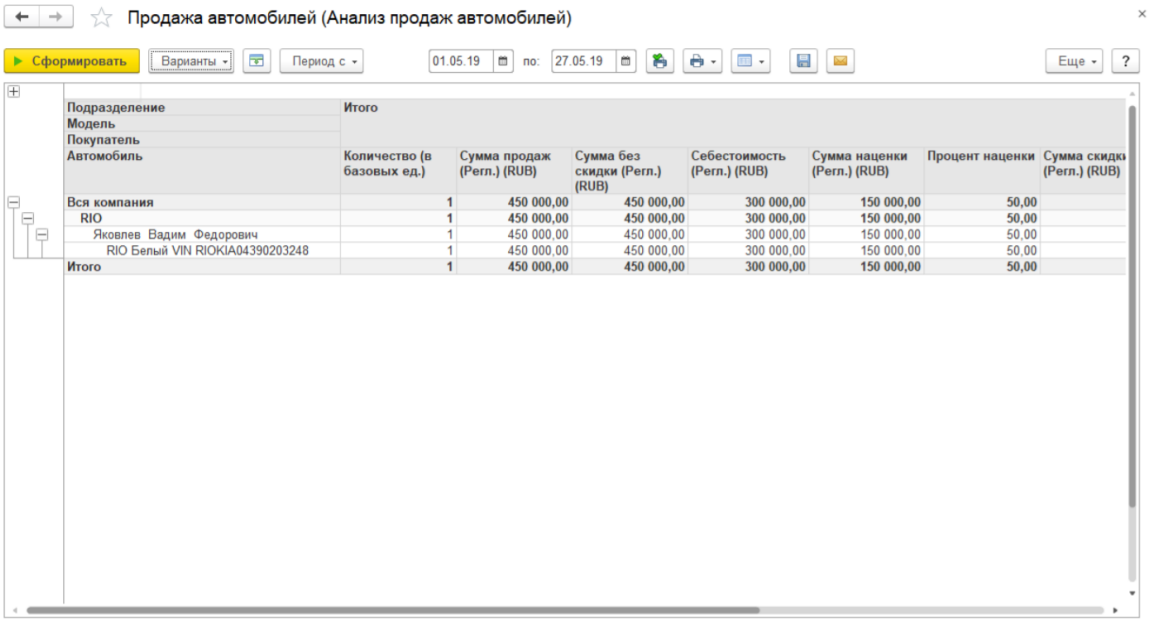


Рис 1.12 Реализация автомобилей

- Аналитические отчеты по продажам автомобилей. Продажи автомобилей можно анализировать при помощи специального отчета в котором можно видеть суммы продажи, себестоимость автомобилей и дополнительного оборудования, сумму и процент скидки, сумму и процент наценки (рис.1.13).



Подразделение / Модель / Покупатель / Автомобиль	Итого						
	Количество (в базовых ед.)	Сумма продаж (Регл.) (RUB)	Сумма без скидки (Регл.) (RUB)	Себестоимость (Регл.) (RUB)	Сумма наценки (Регл.) (RUB)	Процент наценки	Сумма скидки (Регл.) (RUB)
Вся компания	1	450 000,00	450 000,00	300 000,00	150 000,00	50,00	
RIO	1	450 000,00	450 000,00	300 000,00	150 000,00	50,00	
Яковлев Вадим Федорович	1	450 000,00	450 000,00	300 000,00	150 000,00	50,00	
RIO Белый VIN RIOKIA04390203248	1	450 000,00	450 000,00	300 000,00	150 000,00	50,00	
Итого	1	450 000,00	450 000,00	300 000,00	150 000,00	50,00	

Рис 1.13 Анализ продаж автомобилей

- Хранение уникального VIN-номера автомобиля. С помощью VIN-номера можно выгрузить из базы данных все характеристики определенного автомобиля (рис.1.14).

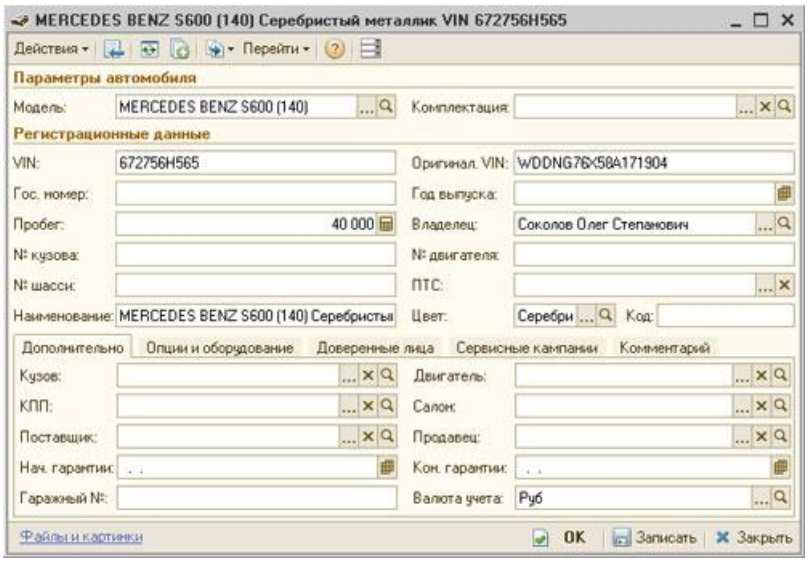


Рис 1.14 Информация об автомобиле

Для всех рассмотренных программных продуктов можно выделить общие функции:

- поиск интересующего автомобиля;
- обширный список характеристик транспортного средства;
- поиск и фильтрация по всем автомобилям;
- возможность связаться с покупателем/продавцом по многим каналам связи.

Особенностью портала Auto.ru является быстрый и удобный поиск по автомобилям от собственников и дилеров. Все характеристики автомобиля находятся в объявлении и при необходимости покупатель всегда может к ним обратиться. Также сервис отсеивает недобросовестных продавцов.

Программный продукт 1С Альфа-Авто позволяет автоматизировать весь процесс управления автосалоном. К основному функционалу относится: ведение рабочих листов, обработка всех типов сообщений, планирование тест-драйвов и анализ продаж.

После проведения анализа предметной области был выделен перечень функций, которые будут реализованы в данной работе:

- Учет автомобилей по VIN-номеру и основным характеристикам.
- Поиск и фильтрация по всем автомобилям.
- Учет и проведение тест-драйва.
- Оформление сделок (договоров) купли-продажи.
- Учет и проведение тест-драйва.
- Анализ проведенных сделок.
- Учет клиентов.

1.2. Концептуальное проектирование

Концептуальное проектирование заключается в формализованном описании предметной области, это описание должно быть таким, чтобы, с одной стороны, можно было проанализировать корректность схемы разрабатываемого проекта БД, с другой стороны, не должно быть привязано к конкретной БД.

Концептуальная модель – модель предметной области, состоящая из перечня взаимосвязанных понятий, используемых для описания этой области, вместе со свойствами и характеристиками, классификацией этих понятий, по типам, ситуациям, признакам в данной области и законов протекания процессов в ней.

Основной целью концептуального проектирования – получение объектов, содержащихся в базе данных.

Любая БД создаётся для решения определённых прикладных задач: в простейшем случае для накопления и выдачи данных, в более сложном случае для решения задач, использующих данные из БД в качестве исходных данных.

Диаграмма вариантов использования является исходным концептуальным представлением системы в процессе ее проектирования и разработки [5].

На диаграмме использования изображаются:

- действующие лица – группы лиц или систем, взаимодействующих с системой;
- варианты использования (прецеденты) — сервисы (функции), которые наша система предоставляет;
- комментарии;
- отношения между элементами диаграммы.

Рассмотрим функции БД для автоматизации работы автосалона. Покупатель заходит в автосалон, где его встречает консультант и уточняет, что посетитель собирается приобрести, также консультант может рассказать о характеристиках определенного автомобиля и записать клиента на тест драйв. Если посетитель собирается купить автомобиль – его провожают к продавцу.

Если клиент пришел для того, чтобы пройти техническое обслуживание, продавец формирует заказ-наряд и передает его механику.

Продавец сообщает информацию о наличии такой комплектации на складе, если такой комплектации нет, то менеджеру передается заказ на определенную комплектацию, и он заказывает ее непосредственно у поставщика. При предзаказе покупатель сразу вносит всю сумму за автомобиль. С продавцом покупатель обсуждает интересующую его модель и комплектацию, после чего они договариваются насчет дополнительного оборудования, которое покупатель хотел бы установить, в том числе: диски, защита картера, защитная пленка и т.п. После чего формируется заказ-наряд для сервиса и продавец выписывает счет.

Механик подготавливает автомобиль к продаже: моет, снимает защитные наклейки, обновляет масла и жидкости, если заказывают, то ставит дополнительное оборудование, проводит техническое обслуживание по заказ-наряду.

Главный менеджер отвечает за закупки автомобилей и дополнительного оборудования. Также менеджер анализирует продажи автомобилей и определенных комплектаций и может проконсультировать клиента по характеристикам автомобиля и наличию комплектаций.

Кладовщик ведет учет автомобилей и дополнительного оборудования на складе, может закупать расходные материалы (масла, тормозные колодки). При недостатке на складе дополнительного оборудования, такого как, сигнализация, дополнительные осветительные устройства, предпусковые подогреватели, кладовщик подает список главному менеджеру, и тот закупает необходимое оборудование.

Когда все технические работы выполнены или автомобиль в нужной комплектации доставлен – продавец выписывает счет.

С проектируемой системой будут взаимодействовать следующие действующие лица:

- консультант,

- продавец,
- механик,
- главный менеджер,
- кладовщик.

Рассмотрим варианты использования каждого из действующих лиц.

Консультанту доступны следующие функции:

1. Запись на тест драйв.
2. Выдача информации о характеристиках и ценах на автомобили.

Продавцу доступны следующие функции:

1. Выдача информации о характеристиках и ценах на автомобили.
2. Выдача информации о наличии комплектаций.
3. Составление комплектации автомобиля с покупателем.
4. Формирование заказ-наряда.
5. Запись на тест-драйв.
6. Формирование договора купли-продажи.
7. Выписывание счета.

Механику доступны следующие функции:

1. Проведение ТО.
2. Обслуживание авто перед продажей.
3. Установка дополнительного оборудования.

Главному менеджеру доступны следующие функции:

1. Выдача информации о характеристиках и ценах на автомобили.
2. Выдача информации о наличии комплектаций.
3. Заказ автомобилей у поставщика.
4. Анализ продаж.
5. Закупка дополнительного оборудования.

Кладовщику доступны следующие функции:

1. Учет комплектаций автомобилей на складе.
2. Выдача информации о наличии комплектаций.
3. Учет дополнительного оборудования на складе.

4. Закупка расходных материалов.
5. Изменение состояния склада после продажи или поставки автомобилей и дополнительного оборудования.

Составим общую диаграмму вариантов использования (рис. 1.15.).

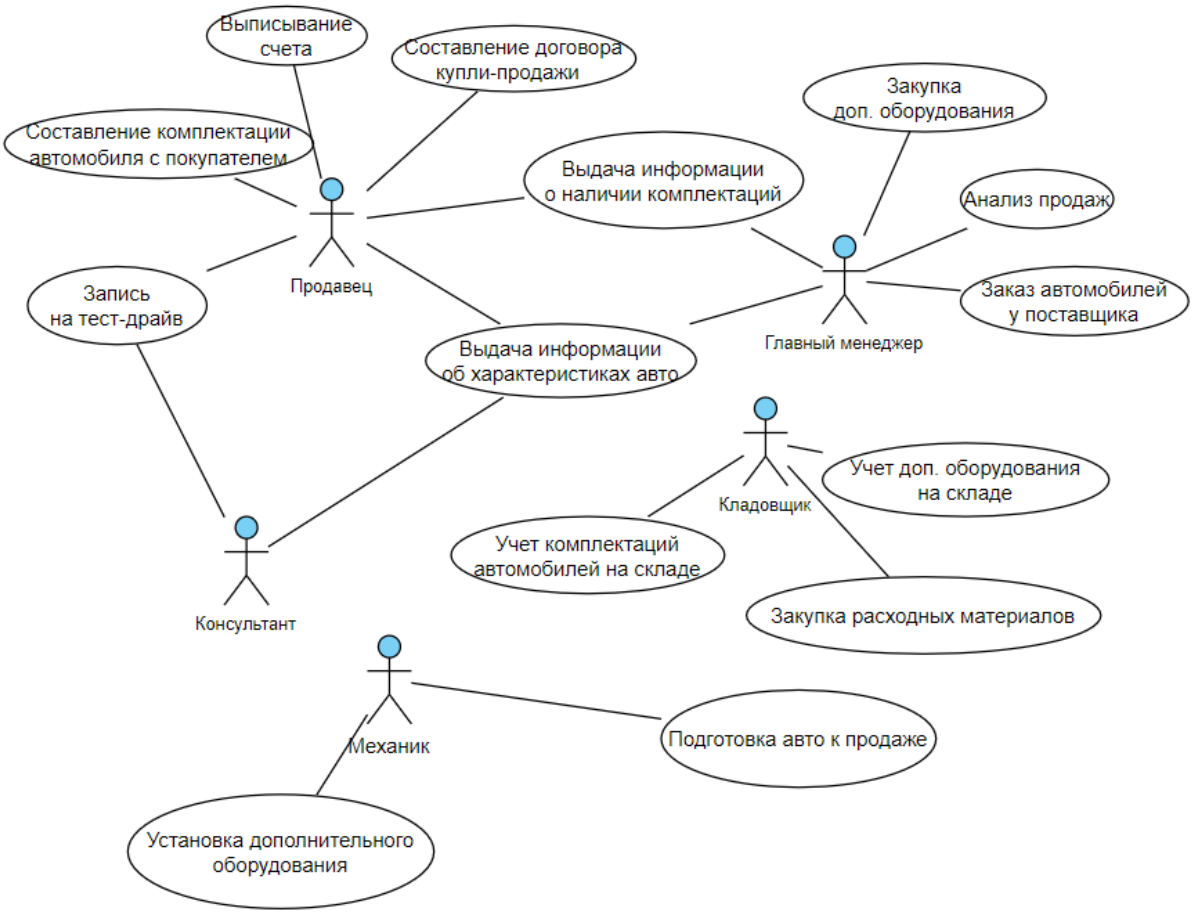


Рис. 1.15. Диаграмма вариантов использования

Заметим, что функции доступные кладовщику схожи с функциями главного менеджера, поэтому в финальной реализации такой роли как кладовщик не будет.

В первом приближении для решения выделенных задач необходимо хранение данных о следующих объектах:

- автомобиль;
- доп. оборудование;
- сотрудник;
- поставщик;
- сделка;

- клиент;
- тест-драйв.

1.3. Логическое проектирование

Логическое проектирование базы данных – это процесс создания модели используемой на предприятии информации на основе выбранной модели организации данных, но без учета типа целевой СУБД и других физических аспектов реализации.

Второй этап проектирования базы данных называется логическим проектированием базы данных. Его цель состоит в создании логической модели данных для исследуемой части предприятия. Концептуальная модель данных, созданная на предыдущем этапе, уточняется и преобразуется в логическую модель данных. Логическая модель данных учитывает особенности выбранной модели организации данных в целевой СУБД (например, реляционная модель).

Если концептуальная модель данных не зависит от любых физических аспектов реализации, то логическая модель данных создается на основе выбранной модели организации данных целевой СУБД. Иначе говоря, на этом этапе уже должно быть известно, какая СУБД будет использоваться в качестве целевой — реляционная, сетевая, иерархическая или объектно-ориентированная. Однако на этом этапе игнорируются все остальные характеристики выбранной СУБД, например, любые особенности физической организации ее структур хранения данных и построения индексов. Построим ER-диаграммы всех сущностей и связей между ними.

Основными понятиями ER-модели являются сущность, связь, атрибут. Сущность – это реальный или представляемый объект, информация о котором должна сохраняться и быть доступна. В диаграммах ER-модели сущность представляется в виде прямоугольника, содержащего имя сущности. При этом имя сущности – это имя типа, а не некоторого конкретного экземпляра этого типа. Для большей выразительности и лучшего понимания имя сущности может сопровождаться примерами конкретных объектов этого типа. Каждый экземпляр сущности должен быть отличим от любого другого экземпляра той

же сущности (это требование в некотором роде аналогично требованию отсутствия кортежей-дубликатов в реляционных таблицах) [4].

Связь – это графически изображаемая ассоциация, устанавливаемая между двумя сущностями. Эта ассоциация всегда является бинарной и может существовать между двумя разными сущностями или между сущностью и ей же самой (рекурсивная связь). В любой связи выделяются два конца (в соответствии с существующей парой связываемых сущностей), на каждом из которых указывается имя конца связи, степень конца связи (сколько экземпляров данной сущности связывается), обязательность связи (т.е. любой ли экземпляр данной сущности должен участвовать в данной связи). Связь между сущностями характеризуется типом связи (1:1, 1:M, M:M) и классом принадлежности (обязательный и необязательный).

Результатом логического проектирования является логическая модель данных, состоящая из ER-диаграммы или диаграммы отношений, а также из реляционной схемы.

На предыдущем этапе были выделены объекты, которые необходимо хранить в базе данных. Эти объекты становятся сущностями при ER моделировании.

Построим ER-диаграммы всех сущностей и связей между ними.

Связь ФОРМИРУЕТ (рис 1.16.) имеет тип 1:M, так как сотрудник может формировать несколько сделок или ни одной. Сделка обязательно формируется одним сотрудником.

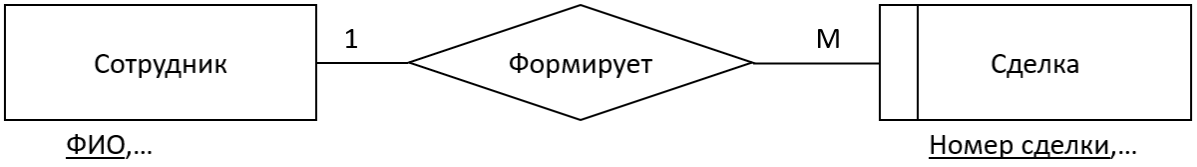


Рис. 1.16. ER-диаграмма связи «Формирует»

Связь УЧАСТВУЕТ (рис 1.17.) имеет тип 1:1, так как в одной сделке может участвовать только один автомобиль. Автомобиль обязательно участвует только в одной сделке или вообще не участвует.

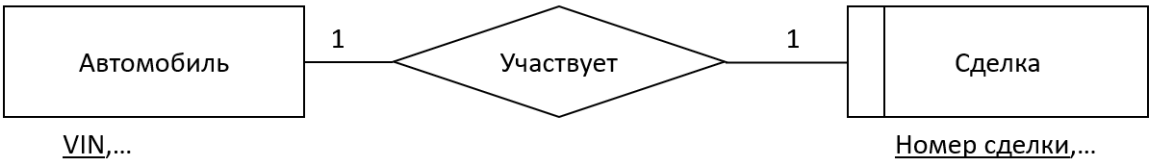


Рис. 1.17. ER-диаграмма связи «Участвует»

Связь ОТНОСИТСЯ (рис 1.18.) имеет тип M:1, так как к сделке обязательно относится одна или несколько деталей оборудования. Деталь обязательно относится только к одной сделке или не относится совсем.

Рис. 1.18. ER-диаграмма связи «Относится»

Связь УЧАСТВУЕТ (рис 1.19.) имеет тип 1:M, так как клиент может участвовать в нескольких сделках. В сделке обязательно участвует один клиент.

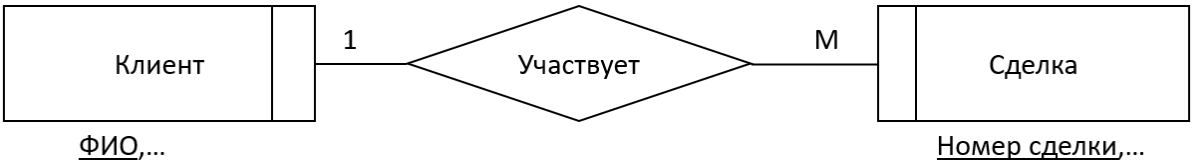


Рис. 1.19. ER-диаграмма связи «Участвует»

Связь ПОСЕЩАЕТ (рис 1.20.) имеет тип 1:M, так как клиент может посетить несколько тест-драйвов или не посещать совсем. Тест драйв обязательно посещает один клиент.

Рис. 1.20. ER-диаграмма связи «Посещает»

Связь ЗАКУПАЕТ (рис 1.21.) имеет тип 1:M, так как сотрудник закупает

несколько деталей. Каждую деталь обязательно закупает один сотрудник.

Рис. 1.21. ER-диаграмма связи «Закупает»

Связь ЗАКУПАЕТ (рис 1.22.) имеет тип 1:M, так как сотрудник закупает несколько автомобилей. Каждый автомобиль обязательно закупает один сотрудник.

Рис. 1.22. ER-диаграмма связи «Закупает»

Связь ПОСТАВЛЯЕТ (рис 1.23.) имеет тип 1:M, так как каждый поставщик может поставлять несколько разных автомобилей. Один автомобиль обязательно поставлен одним поставщиком.

Рис. 1.23. ER-диаграмма связи «Поставляет 1»

Связь ПОСТАВЛЯЕТ (рис 1.24.) имеет тип 1:M, так как каждый поставщик может поставлять несколько разных деталей. Одна деталь обязательно поставлена одним поставщиком.

Рис. 1.24. ER-диаграмма связи «Поставляет 2»

Связь УЧАСТВУЕТ (рис 1.25.) имеет тип 1:M, так как каждый автомобиль может участвовать в нескольких или ни одном тест драйве. В тест-драйве обязательно участвует один автомобиль.

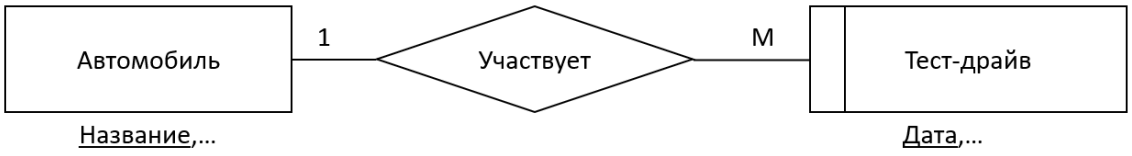


Рис. 1.25. ER-диаграмма связи «Участвует»

Сформируем набор предварительных отношений с указанием предполагаемого первичного ключа для каждого отношения.

Связь ФОРМИРУЕТ удовлетворяет условиям правила 4, в соответствии с которым получаем два отношения:

1. Сотрудник (ФИО, …)
2. Сделка (Номер сделки, ФИО сотрудника, …)

Связь УЧАСТВУЕТ удовлетворяет условиям правила 2, в соответствии с которым получаем два отношения:

1. Автомобиль (VIN, …)
2. Сделка (Номер сделки, VIN автомобиля, …)

Связь ОТНОСИТСЯ удовлетворяет условиям правила 4, в соответствии с которым получаем два отношения:

1. Доп. оборудование (Номер детали, …)
2. Сделка (Номер сделки, …)
3. Доп. оборудование сделки (Номер сделки, Номер детали)

Связь УЧАСТВУЕТ удовлетворяет условиям правила 4, в соответствии с которым получаем два отношения:

1. Клиент (ФИО, …)
2. Сделка (Номер сделки, ФИО клиента, …)

Связь ПОСЕЩАЕТ удовлетворяет условиям правила 4, в соответствии с которым получаем два отношения:

1. Клиент (ФИО, …)

2. Тест-драйв (Номер тест-драйва, ФИО клиента, …)

Связь ЗАКУПАЕТ удовлетворяет условиям правила 4, в соответствии с которым получаем два отношения:

1. Сотрудник (ФИО, …)
2. Автомобиль (VIN, Закупщик…)

Связь ЗАКУПАЕТ удовлетворяет условиям правила 4, в соответствии с которым получаем два отношения:

1. Сотрудник (ФИО, …)
2. Доп. оборудование (Номер детали, Закупщик, …)

Связь ПОСТАВЛЯЕТ удовлетворяет условиям правила 4, в соответствии с которым получаем два отношения:

1. Поставщик (Название, …)
2. Автомобиль (VIN, Поставщик, …)

Связь ПОСТАВЛЯЕТ удовлетворяет условиям правила 4, в соответствии с которым получаем два отношения:

1. Поставщик (Название, …)
2. Доп. оборудование (Номер детали, Поставщик, …)

Связь УЧАСТВУЕТ удовлетворяет условиям правила 4, в соответствии с которым получаем два отношения:

1. Автомобиль (VIN, …)
2. Тест-драйв (Номер тест-драйва, VIN автомобиля, …)

Добавим не ключевые атрибуты в каждое из предварительных отношений с условием, чтобы отношения отвечали требованиям третьей нормальной формы.

Схема полученной БД:

1. Автомобиль (VIN, Поставщик, Закупщик, Марка, Модель, Год выпуска, Тип кузова, Цвет, Материал салона, Коробка передач, Мощность, Стоимость)
2. Сотрудник (ФИО, Номер телефона, Должность)
3. Клиент (ФИО, Номер телефона, Паспортные данные)

4. Сделка (Номер сделки, ФИО сотрудника, ФИО клиента, VIN-автомобиля, Дата, Стоимость)
5. Тест-драйв (Номер тест-драйва, Дата, Время, VIN-автомобиля, ФИО клиента)
6. Поставщик (Название, Номер телефона)
7. Доп. оборудование (Номер детали, Название, Закупщик, Поставщик)
8. Доп. оборудование сделки (Номер сделки, Номер детали)

1.4. Физическое проектирование

Физическое проектирование – процесс подготовки описания реализации базы данных на вторичных запоминающих устройствах; на этом этапе рассматриваются основные отношения, организация файлов и индексов, предназначенных для обеспечения эффективного доступа к данным, а также все связанные с этим ограничения целостности.

Физическое проектирование является третьим и последним этапом создания проекта базы данных, при выполнении которого проектировщик принимает решения о способах реализации разрабатываемой базы данных. Во время предыдущего этапа проектирования была определена логическая структура базы данных, состоящая из ER-диаграммы или диаграммы отношений, а также из реляционной схемы (которая описывает отношения и ограничения в рассматриваемой прикладной области). Она же является исходной информацией.

Между логическим и физическим проектированием существует постоянная обратная связь, так как решения, принимаемые на этапе физического проектирования с целью повышения производительности системы, способны повлиять на структуру логической модели данных.

Целью данного этапа является реализация спроектированной БД на конкретной СУБД и обеспечение выполнения всех требований к ней.

Для физического проектирования базы данных автосалона была выбрана СУБД SQL. MS SQL Server Management Studio (SSMS) – интегрированная среда для управления любой инфраструктурой SQL, разработанная корпорацией Microsoft. SSMS предоставляет средства для настройки, наблюдения и администрирования экземпляров SQL Server и баз данных. С помощью SSMS можно развертывать, отслеживать и обновлять компоненты уровня данных, создавать запросы и скрипты.

SQL Server Management Studio подходит для создания запросов к базам данных и хранилищам данных, их проектирования и управления ими, где бы они ни находились: на локальном компьютере или в облаке.

Исходя из результатов прошлого этапа, для полученных отношений будут сформированы таблицы.

Таблица «Автомобиль» (см. табл. 1) содержит данные о всех автомобилях, находящихся в автосалоне.

Таблица 1

Таблица «Автомобиль»

Столбец	Тип данных	Нуль?	Ключ	Значение по умолчанию	Ограничение	Ссылка
VIN	Строка	Нет	Первичный			
Поставщик	Строка	Нет				Название в Поставщик
Закупщик	Строка	Нет				ФИО в Сотрудник
Марка	Строка	Нет				
Модель	Строка	Нет				
Год выпуска	Целый	Нет			>1950	
Тип кузова	Строка	Нет		Седан		
Цвет	Строка	Нет		Черный		
Материал салона	Строка	Нет				
Коробка передач	Строка	Нет		Автоматическая	Автоматическая, Механическая	
Тип двигателя	Строка	Нет		Бензиновый	Бензиновый, Дизельный, Электрический	
Мощность	Целый	Нет				
Стоимость	Денежный	Нет			>0	

Таблица «Сотрудник» (см. табл. 2) содержит данные о сотрудниках автосалона.

Таблица 2

Таблица «Сотрудник»

Столбец	Тип данных	Нуль?	Ключ	Значение по умолчанию	Ограничение	Ссылка
ФИО	Строка	Нет	Первичный			
Номер телефона	Строка	Нет				
Должность	Строка	Нет				

Таблица «Клиент» (см. табл. 3) содержит данные о клиентах автосалона.

Таблица 3

Таблица «Клиент»

Столбец	Тип данных	Нуль?	Ключ	Значение по умолчанию	Ограничение	Ссылка
ФИО	Строка	Нет	Первичный			
Номер телефона	Строка	Нет				
Паспортные данные	Строка	Нет	Уникальный			

Таблица «Сделка» (см. табл. 4) содержит данные о сделках, проведенных в автосалоне.

Таблица 4

Таблица «Сделка»

Столбец	Тип данных	Нуль?	Ключ	Значение по умолчанию	Ограничение	Ссылка
Номер сделки	Целый	Нет	Первичный			
ФИО сотрудника	Строка	Нет				ФИО в Сотрудник
ФИО клиента	Строка	Нет				ФИО в Клиент
VIN автомобиля	Строка	Нет				VIN в Автомобиль
Дата	Дата	Нет		Текущая дата	= Текущая дата	
Стоимость	Денежный	Нет			>0	

Таблица «Тест-драйв» (см. табл. 5) содержит данные о тест-драйвах.

Таблица 5

Таблица «Тест-драйв»

Столбец	Тип данных	Нуль?	Ключ	Значение по умолчанию	Ограничение	Ссылка
Номер тест-драйва	Целый	Нет	Первичный			
Дата	Дата	Нет		Текущая дата	>=Текущая дата	
Время	Время	Нет				
VIN автомобиля	Строка	Нет				VIN в Автомобиль
ФИО клиента	Строка	Нет				ФИО в Клиент

Таблица «Поставщик» (см. табл. 6) содержит данные о поставщиках.

Таблица 6

Таблица «Поставщик»

Столбец	Тип данных	Нуль?	Ключ	Значение по умолчанию	Ограничение	Ссылка
Название	Строка	Нет	Первичный			
Номер телефона	Строка	Нет				

Таблица «Доп. оборудование» (см. табл. 7) содержит данные о дополнительном оборудовании.

Таблица 7

Таблица «Доп. оборудование»

Столбец	Тип данных	Нуль?	Ключ	Значение по умолчанию	Ограничение	Ссылка
Номер детали	Целое	Нет	Первичный			
Название	Строка	Нет				
Закупщик	Строка	Нет				ФИО в Сотрудник
Поставщик	Строка	Нет				Название в Поставщик

Таблица «Доп. оборудование сделки» (см. табл. 8) содержит данные о дополнительном оборудовании участвующем в сделках.

Таблица 8

Таблица «Доп. оборудование сделки»

Столбец	Тип данных	Нуль?	Ключ	Значение по умолчанию	Ограничение	Ссылка
Номер сделки	Целое	Нет	Первичный			Номер сделки в Сделка
Номер детали	Целое	Нет				Номер детали в Доп. оборудование

Заполнение таблиц тестовыми данными приведено в Приложении Б.

На основе созданных таблиц можно составить схему базы данных автосалона (рис. 4.1).

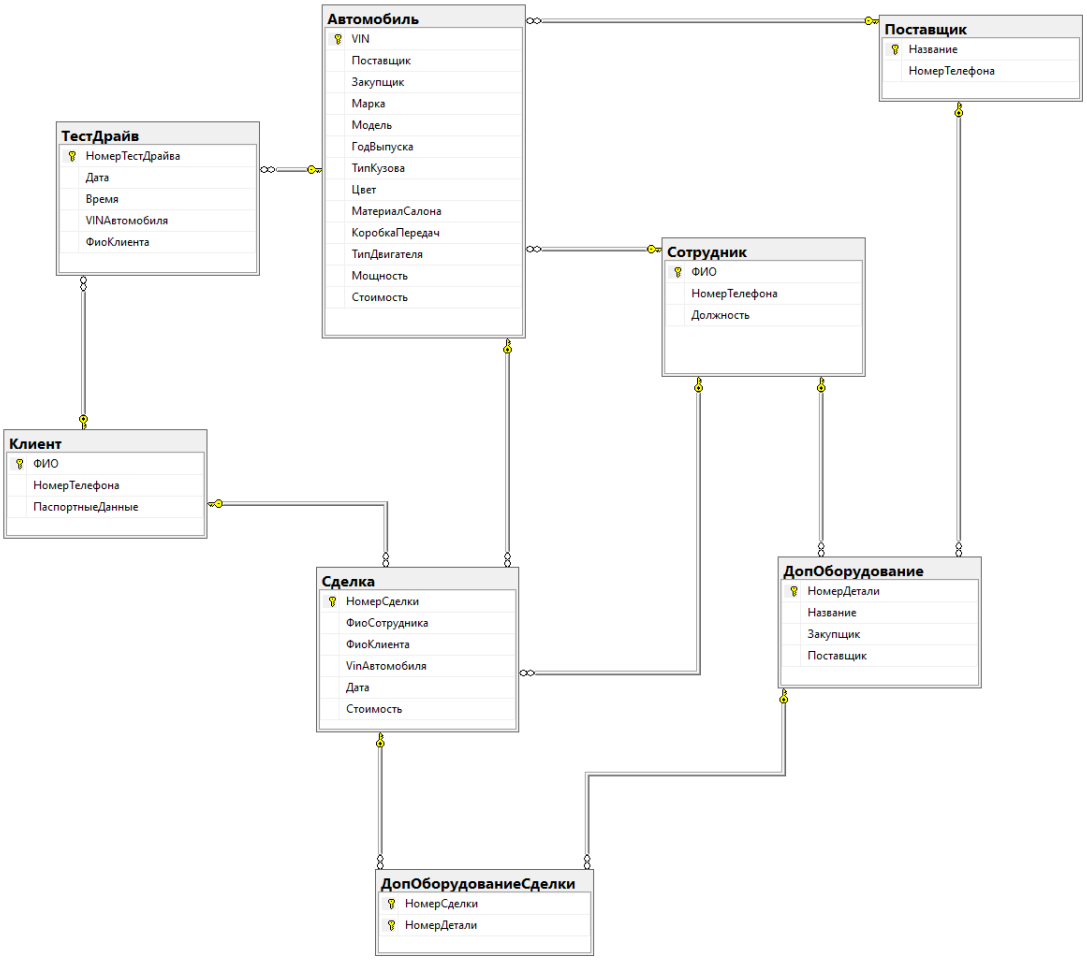


Рис. 1.26. Схема базы данных автосалона

У таблиц также есть собственные функции и процедуры.

У таблицы Автомобиль есть 2 функции: InsertCar, PriceByVin (Приложение Б, п.1, п.2), а также 3 табличных функции CountOfCarsNotInDealByMark, CarsNotInDeal и ModelsByMark (Приложение Б,

п.3, п.4, п.5). Процедура InsertCar нужна для добавление в БД нового автомобиля. Входными параметрами этой процедуры являются VIN номер автомобиля, название поставщика, ФИО закупщика, марка, модель, год выпуска, тип кузова, цвет, материал салона, тип коробки передач, тип двигателя, мощность и цена. Внутри процедуры входные данные проверяются на корректность, если все тесты пройдены, то запись добавляется в таблицу и процедура возвращает номер ошибки 0, иначе добавления не происходит и процедура возвращает номер ошибки больше нуля, который потом можно расшифровать. Процедура PriceByVin возвращает цену автомобиля с указанным VIN номером. Входным параметром этой функции является VIN автомобиля, а выходным его цена. Табличная функция CountOfCarsNotInDealByMark возвращает количество оставшихся на складе автомобилей по маркам. Входных параметров у этой функции нет. Вторая табличная функция CarsNotInDeal возварщает список VIN номеров автомобилей которые не участвуют в сделках и следовательно могут быть проданы. Входных параметров у данный функции нет. Последняя табличная функция ModelsByMark возвращает списков моделей определенной марки. Входным параметром ModelsByMark является марка автомобиля, модели которой необходимо получить.

У таблицы Поставщик есть 1 процедура InsertPost (Приложение Б, п.6). Эта процедура используется для добавления в БД нового поставщика. Входные параметры: название поставщика и номер телефона. Во время работы процедура проверяет нет ли в БД поставщика с таким названием и номером телефона, если есть, то добавление не происходит и процедура возвращает номер ошибки больше нуля. Если такого поставщика нет, то происходит добавление и процедура возвращает номер ошибки 0.

У таблицы ДопОборудование есть 1 процедура InsertDetail (Приложение Б, п.7) и 3 табличные функции DetailsOfDeal, DetailNotInDeal, CountOfDetailsNotInDealByName (Приложение Б, п.8, п.9, п.10). Процедура InsertDetail используется для добавления в БД новой детали. Входные параметры: номер детали, название, ФИО закупщика и название поставщика. Во

время работы процедура проверяет нет ли в уже в БД детали с таким номером, если есть, то добавление не происходит и процедура возвращает номер ошибки больше нуля. Если такого поставщика нет, то происходит добавление и процедура возвращает номер ошибки 0. Табличная функция DetailsOfDeal возваращает все характеристики деталей которые находятся в определенной сделке. Входной параметр – номер сделки. Табличная функция DetailNotInDeal возвращает список деталей которые не участвуют в сделках. Входных параметров у этой табличной функции нет. Табличная функция CountOfDetailsNotInDealByName возвращает оставшееся количество деталей на складе группирую по названиям. Входных параметров у этой табличной функции нет.

У таблицы Сотрудник есть 1 процедура InsertSotr (Приложение Б, п.11) и 2 табличных функции ManagersList, ManagersAndStorekeepers (Приложение Б, п.12, п.13).

Процедура InsertSotr используется для добавления в БД нового сотрудника. Входные параметры: ФИО сотрудника, номер телефона, должность. Во время работы процедура проверяет нет ли в БД сотрудника с такой ФИО или таким номером телефона, если есть, то добавление не происходит и процедура возвращает номер ошибки больше нуля. Если сотрудника с такими данными нет, то происходит добавление и процедура возвращает номер ошибки 0.

Функция ManagersList возвращает список менеджеров. Входных параметров нет.

Табличная функция ManagersAndStorekeepers подобна функции ManagersList, главное отличие что вторая функция помимо менеджеров возвращает ещё и продавцов.

У таблицы ДопОборудованиеСделки есть 1 процедура InsertDealDetail (Приложение Б п.14). Эта процедура используется для добавления в сделку деталей. Входные параметры: номер сделки и номер детали. Во время работы процедура добавляет в таблицу ДопОборудованиеСделки новую запись.

У таблицы Клиент есть 1 процедура InsertClient (Приложение Б п.15). Эта процедура используется для добавления в БД нового клиента. Входные параметры: ФИО клиента, номер телефона, серия и номер паспорта. Во время работы процедура проверяет нет ли в БД клиента с таким именем, номером телефона или паспортом, если есть, то добавление не происходит и процедура возвращает номер ошибки больше нуля. Если такого клиента нет, то происходит добавление и процедура возвращает номер ошибки 0.

У таблицы ТестДрайв есть 1 процедура InsertTestDrive (Приложение Б п.16). Эта процедура используется для добавления в БД записи о новом тест-драйве. Входные параметры: дата тест-драйва, время, VIN номер автомобиля, ФИО клиента. Во время работы процедура проверяет нет ли записи на это же время и дату с этим автомобилем, а также существование такого автомобиля и клиента. Если все условия выполнены, то происходит добавление и процедура возвращает номер ошибки 0.

У таблицы Сделка есть 1 процедура InsertDeal (Приложение Б п.17), 1 табличная функция SellsByMonth (Приложение Б п.18) и триггер DeleteDetailInDeal (Приложение Б п.19).

Процедура InsertDeal используется для добавления в БД записи о новой сделке. Входные параметры: ФИО сотрудника, ФИО клиента, VIN автомобиля, дата сделки, итоговая сумма сделки. Во время работы процедура существуют ли данные сотрудник, клиент и автомобиль. Также не допускается чтобы итоговая сумма сделки была меньше стоимости автомобиля из таблицы Автомобиль. На место номера сделки ставится максимальный номер из таблицы Сделка, к которому прибавляется единица. Если все условия выполнены, то происходит добавление и процедура возвращает номер ошибки 0.

Табличная функция SellsByMonth подсчитывает суммарную сумму сделок за определенный месяц группируя по сотрудникам. На вход идет месяц, за который необходимо получить статистику.

Триггер DeleteDeatilInDeal при удалении записи из таблицы «Сделка» удаляет все связанные с этой сделкой детали из таблицы «ДопОбоорудованиеСделки».

ГЛАВА 2. Описание функционирования базы данных

2.1. Назначение и перечень функций базы данных

Основной задачей курсового проекта является реализация удобного и понятного интерфейса для поиска, удаления, добавления и изменения информации в БД. Программный продукт должен автоматизировать основную часть работы автосалона. Работать с базой могут все сотрудники, однако функционал разделен по должностям, для предотвращения деятельности сотрудников вне своих областей ответственности. При вводе некорректных значений интерфейс должен выводить ошибку и её описание.

Менеджеры могут использовать такие функции БД как:

- Добавление, удаление и изменение данных об автомобилях, а также поиск автомобиля по его VIN номеру;
- Добавление, удаление и изменение данных о поставщиках, а также поиск поставщика по названию;
- Добавление, удаление и изменение данных о сотрудниках, а также поиск сотрудника по ФИО либо по должности;
- Получение информации о результативности продаж сотрудников по месяцу;
- Добавление, удаление и изменение данных о сотрудниках, а также поиск сотрудника по ФИО либо по должности;
- Добавление, удаление и изменение данных о дополнительном оборудовании, а также поиск деталей по названию;
- Получение информации о остатках автомобилей на складе по маркам, а также информации об остатках дополнительного оборудования по названию.

Продавцы могут использовать такие функции БД как:

- Подбор автомобиля по всем характеристикам от марки до цены;
- Добавление и изменение данных о клиентах, а также поиск клиента то ФИО либо по номеру телефона;

- Добавление, удаление и изменение данных о сделках, а также поиск сделки по ФИО клиента с которым она была заключена;
- Добавление и удаление дополнительного оборудования уже существующей сделке.

Консультанты могут использовать такие функции БД как:

- Добавление, удаление и изменение данных о тест-драйвах, а также поиск определенной записи на тест-драйв по дате либо по ФИО клиента;
- Подбор автомобиля по всем характеристикам от марки до цены;
- Добавление и изменение данных о клиентах, а также поиск клиента то ФИО либо по номеру телефона;
- Подбор детали по названию и поставщику.

Механики могут использовать такие функции БД как:

- Информация об дополнительном оборудовании на автомобиле который находится в сделке. Поиск возможен как по VIN номеру автомобиля, так и по номеру сделки.

Для выполнения программной части курсового проекта были использованы: СУБД Microsoft SQL Server Management Studio и язык программирования C# [6]. Средой разработки была выбрана Microsoft Visual Studio.

2.2. Описание работы с базой данных

В начале работы для доступа к функционалу сотрудник должен выбрать роль, под которой он заходит в систему. Для менеджеров и продавцов предусмотрен выбор ФИО, так как это информация используется для ввода данных. После выбора роли сотрудник нажимает кнопку «Войти» (Рис. 2.1., приложение В п.1).

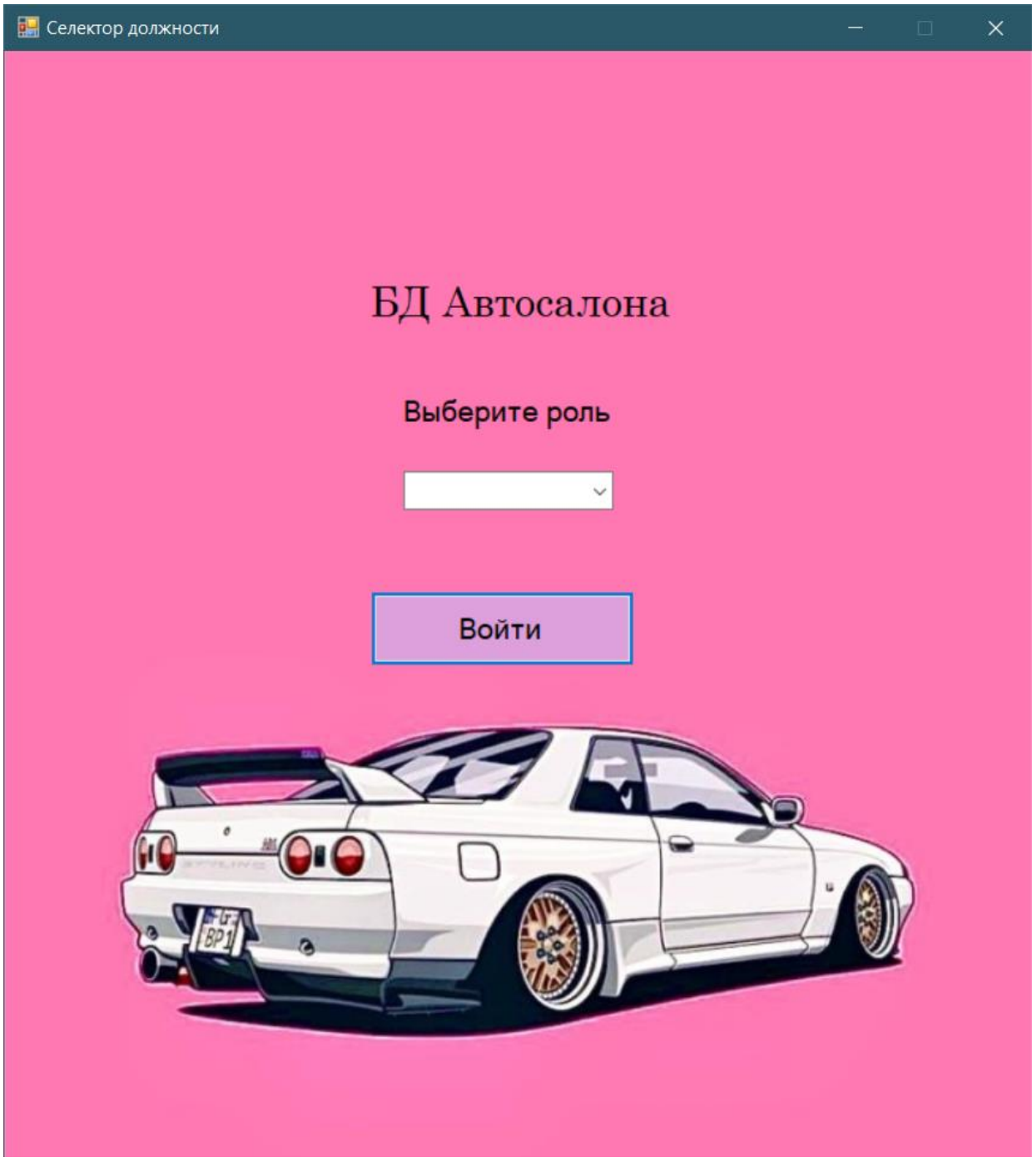


Рис. 2.1. Начальная форма

После нажатия открывается форма соответствующая выбранной роли.

Рассмотрим все роли и их формы.

Выбрав роль менеджера, сотрудник попадает на домашнюю страницу менеджера. Изначально открыта вкладка «Автомобиль» (Рис. 2.2., приложение В п.2), на ней сотрудник может добавить новый автомобиль в базу, отредактировать цену уже существующего автомобиля и удалить автомобиль из базы. Программа учитывает, что VIN номер автомобиля состоит только из 17 символов, поэтому при введении меньшего или большего количества символов высвечивается соответствующая ошибка. Также стоит запрет на год выпуска ранее 1950 года и цену меньше 0. В поле закупщик автоматически вставляется ФИО сотрудника. При попытке удаления автомобиля участвующего в сделке форма выведет ошибку.

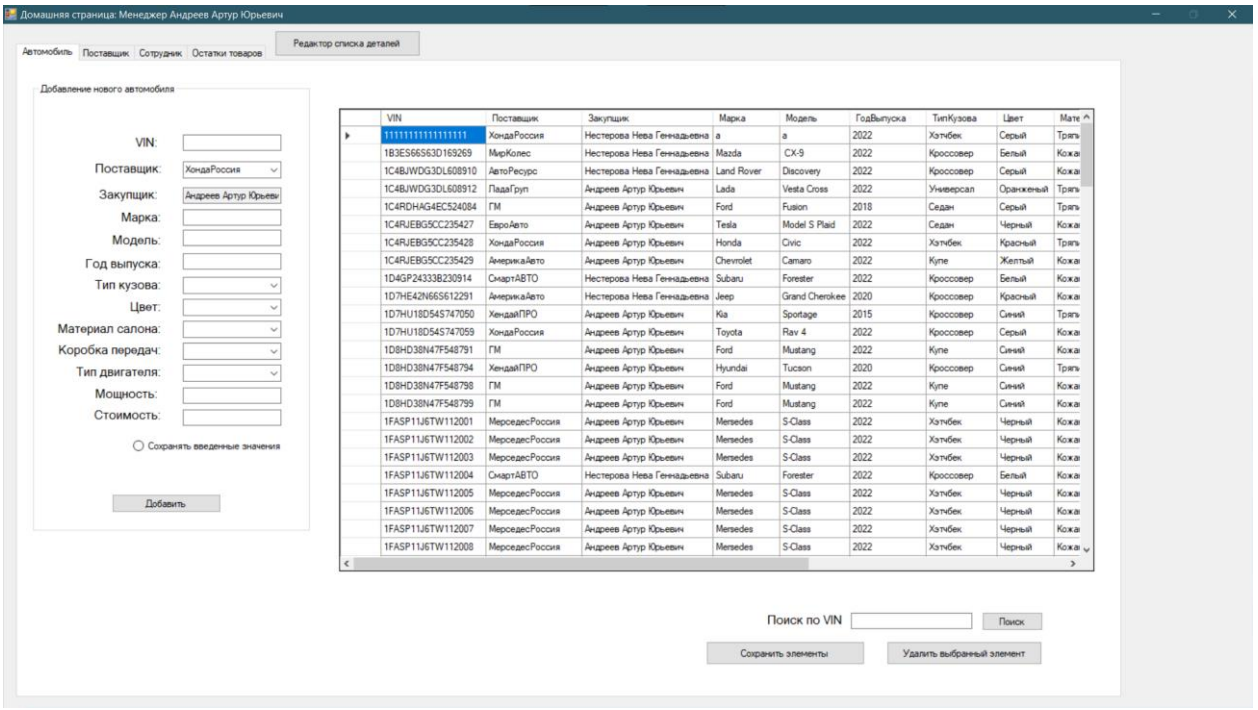


Рис. 2.2. Форма «Домашняя страница: Менеджер» вкладка «Автомобиль»

Перейдем на вкладку «Поставщик» (Рис. 2.3., приложение В п.2), на ней сотрудник может редактировать список поставщиков, а также добавлять новых. На этой вкладке реализован поиск поставщиков по названию. На поле номер телефона стоит ограничение по количеству и типу введенных символов – только 12 символов и все цифры.

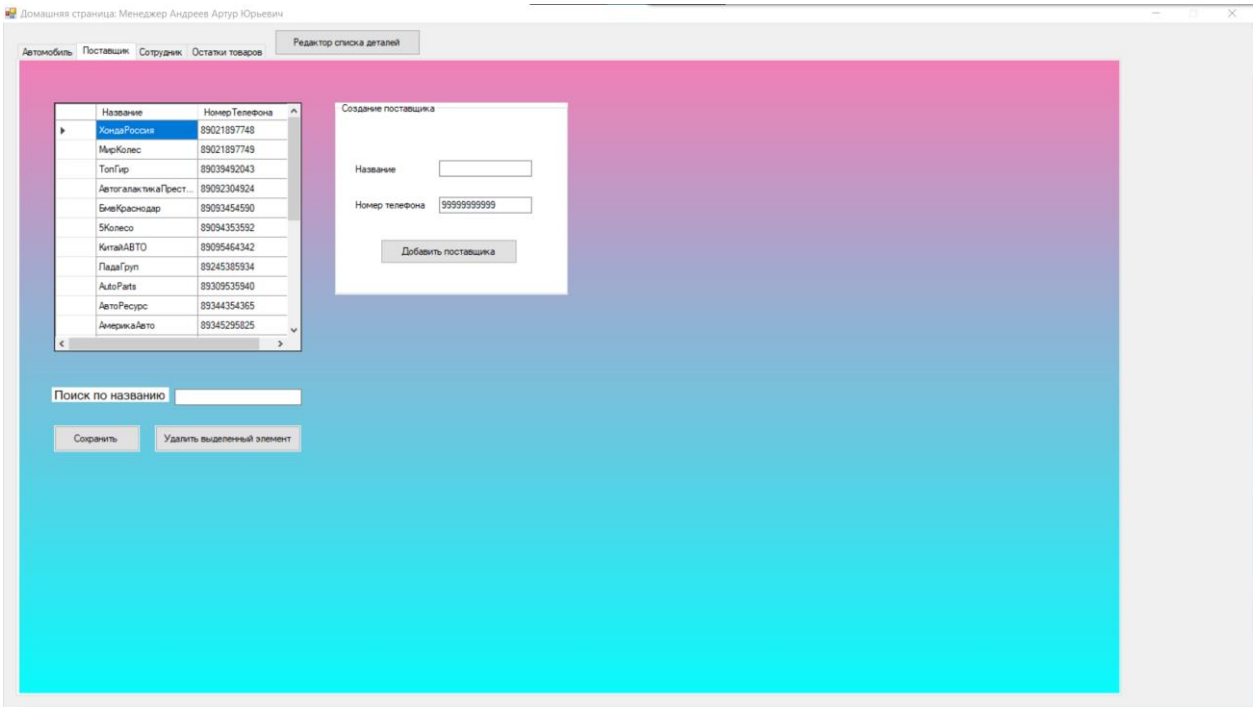


Рис. 2.3. Форма «Домашняя страница: Менеджер» вкладка «Поставщик»

Перейдем на вкладку «Сотрудник» (Рис. 2.4., приложение В п.2). На ней менеджер может редактировать список сотрудников автосалона и добавлять новых. Также реализована функция поиска сотрудников по ФИО и по должности. На форме есть таблица результативности продаж сотрудников по месяцу, для того чтобы менеджер мог отслеживать эффективность сотрудников. Поля для введения данных отслеживают введение корректных значений, в противном случае они выводят ошибку.

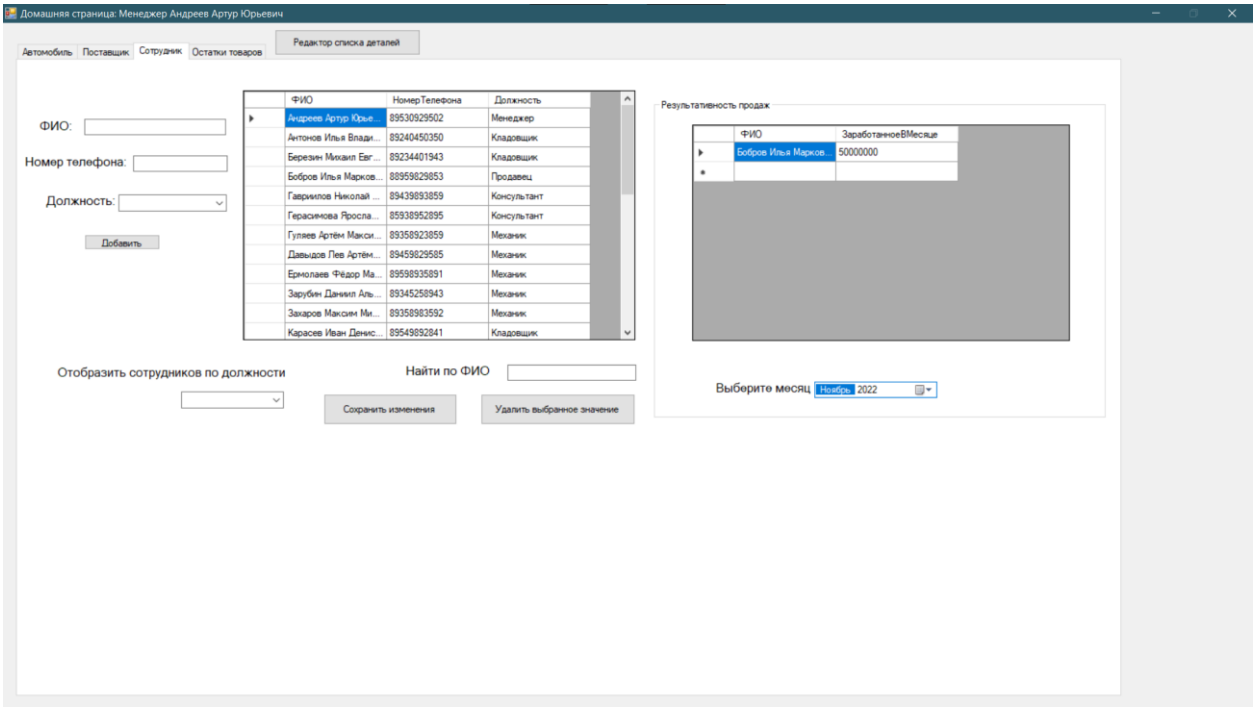


Рис. 2.4. Форма «Домашняя страница: Менеджер» вкладка «Сотрудник»

Вкладка «Остатки товаров» (Рис. 2.5., приложение В п.2) предназначена для вывода информации об остатках автомобилей и дополнительного оборудования на складе. Автомобили сгруппированы по маркам, дополнительное оборудование по названию. С помощью данный формы менеджер может планировать закупки недостающего оборудования.

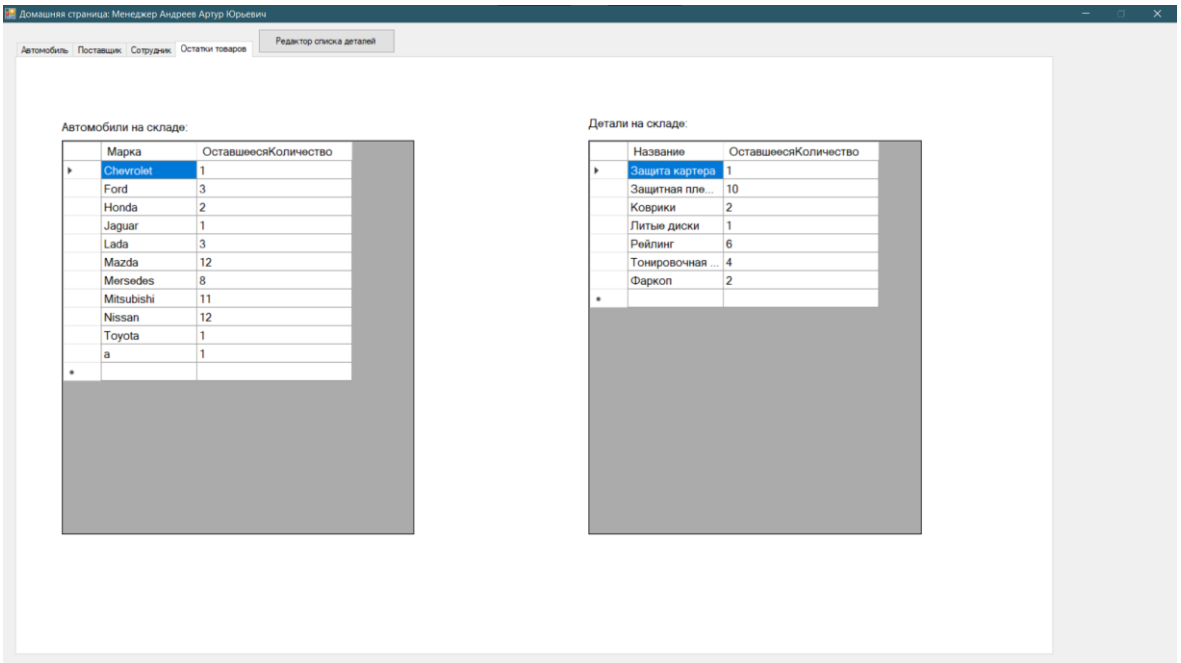


Рис. 2.5. Форма «Менеджер» вкладка «Остатки товаров»

При нажатии кнопки «Редактор списка деталей» откроется форма позволяющая сотруднику редактировать список деталей в базе данных (Рис. 2.6., приложение В п.2). На форме расположены 2 таблицы, одна из которых отображает все детали проходящие через автосалон, а вторая только детали не принадлежащие к сделкам. Поля «Номер детали» и «Закупщик» заполняются автоматически.

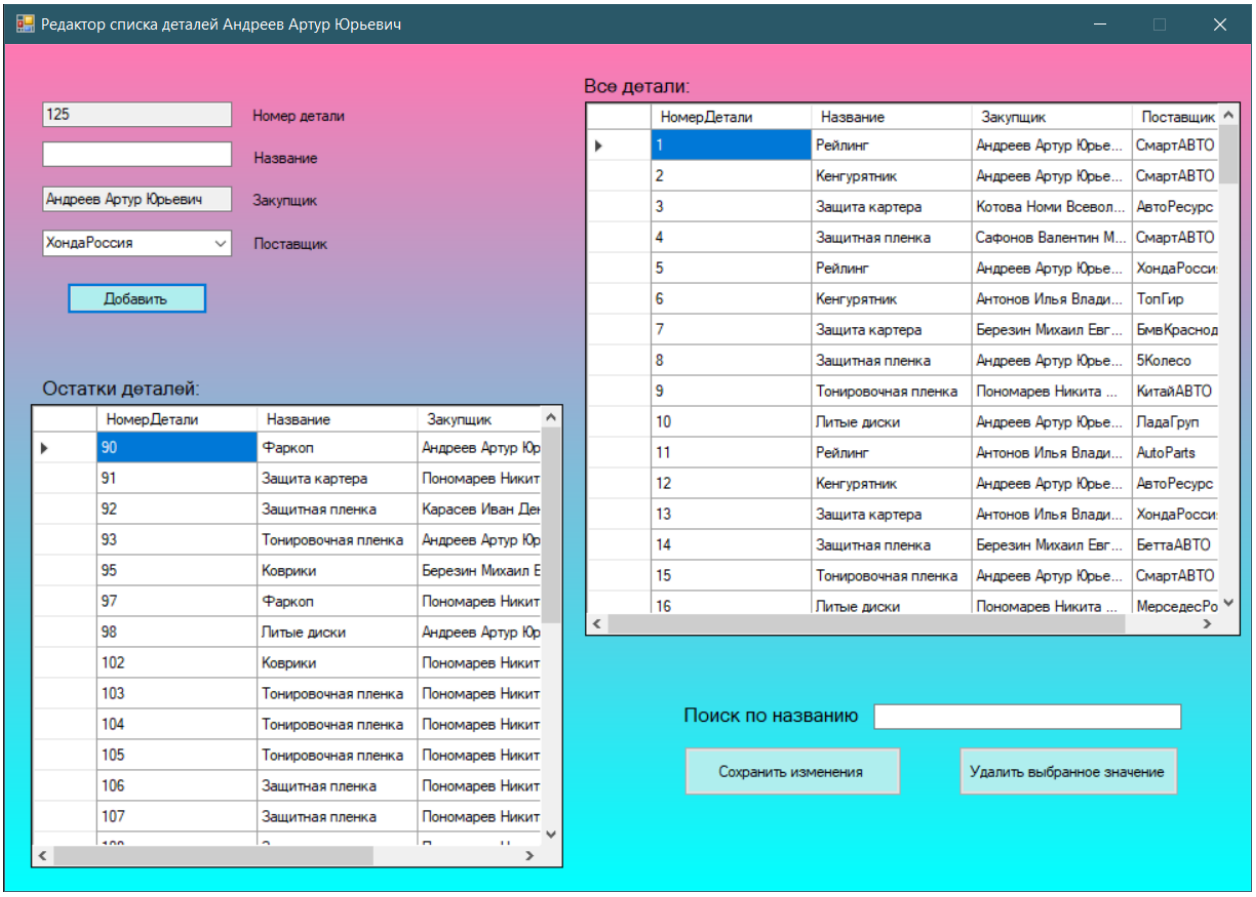


Рис. 2.6. Форма «Редактор списка деталей»

На этом функции доступные менеджерам заканчиваются.

Перейдем к функционалу доступному консультантам. При входе в роль «Консультант» перед сотрудником открывается домашняя страница (Рис. 2.7., приложение В п.3). В ней есть 4 кнопки: «Запись на тест драйв», «Подобрать автомобиль», «Подобрать детали», «Создать клиента». Каждая кнопка открывает соответствующую форму, реализующую описанные функции.

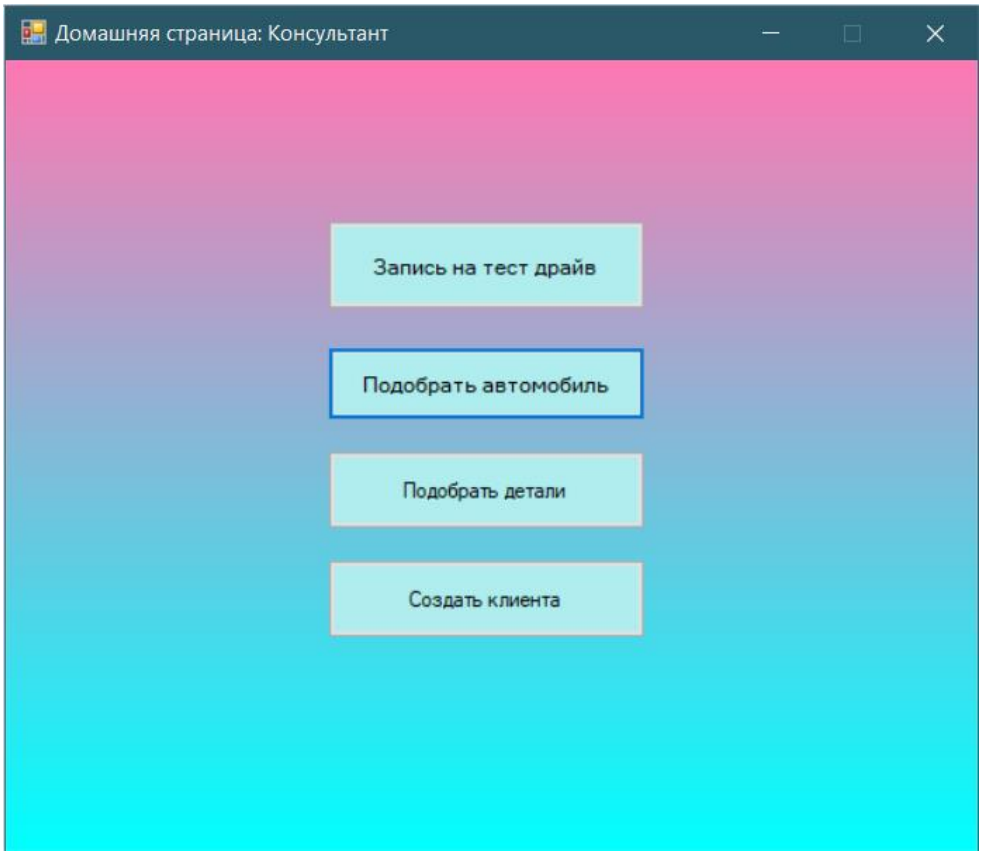


Рис. 2.7. Форма «Домашняя страница: Консультант»

При нажатии кнопки «Запись на тест драйв» открывается форма позволяющая сотруднику редактировать список тест-драйвов (Рис. 2.8., приложение В п.4). Также в ней доступен поиск записей по ФИО клиента и по дате тест-драйва. На форме расположены кнопки открытия форм подбора автомобиля создания клиента для удобства записи клиента на тест драйв.

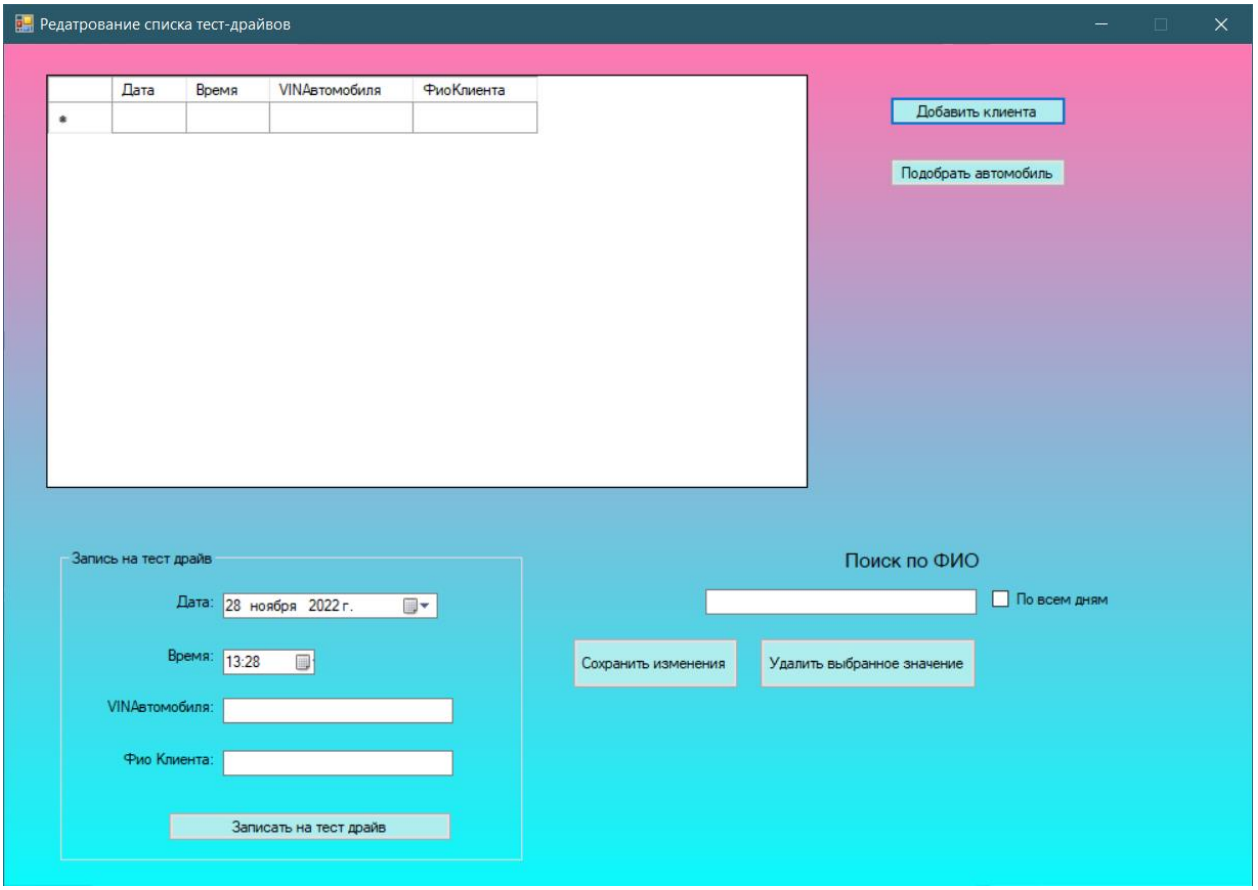


Рис. 2.8. Форма «Редактирование списка тест-драйвов»

Форма подбора автомобиля (Рис. 2.9., приложение В п.5) предоставляет сотруднику возможность находить автомобили на складе фильтруя их по всем критериям начиная маркой автомобиля и заканчивая желаемой стоимостью. Возможность редактирования табличной части заблокирована.

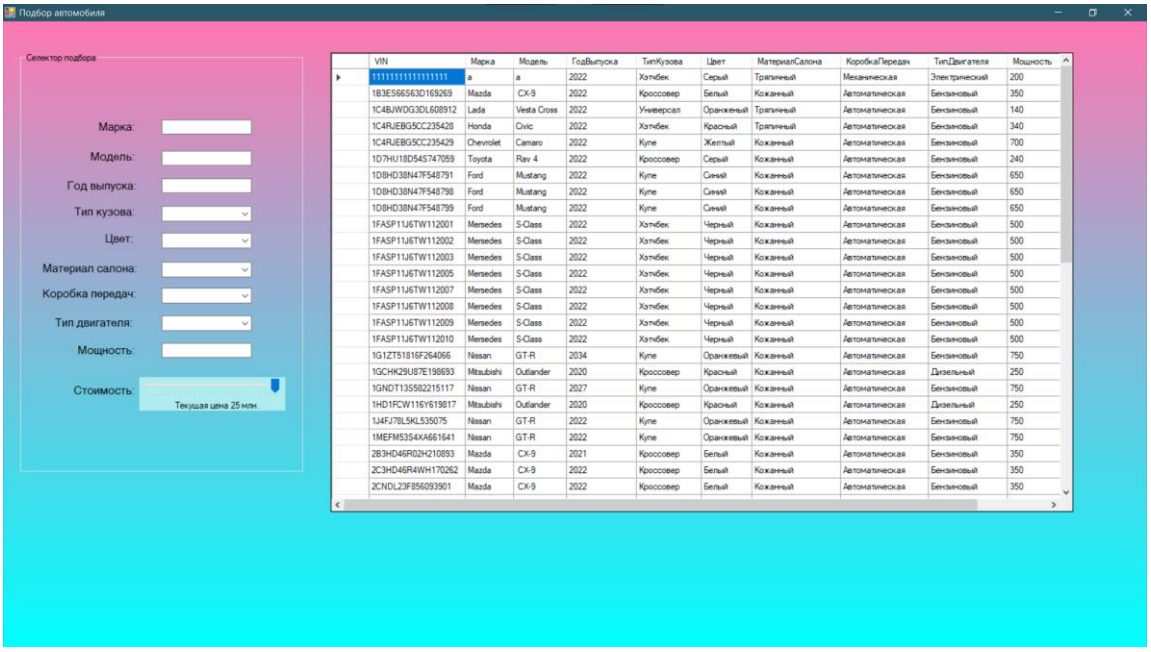


Рис. 2.9. Форма «Подбор автомобиля»

Форма подбора деталей (Рис. 2.10., приложение В п.6) предоставляет сотруднику возможность находить детали на складе фильтруя их по названию либо по поставщику. Возможность редактирования табличной части заблокирована.

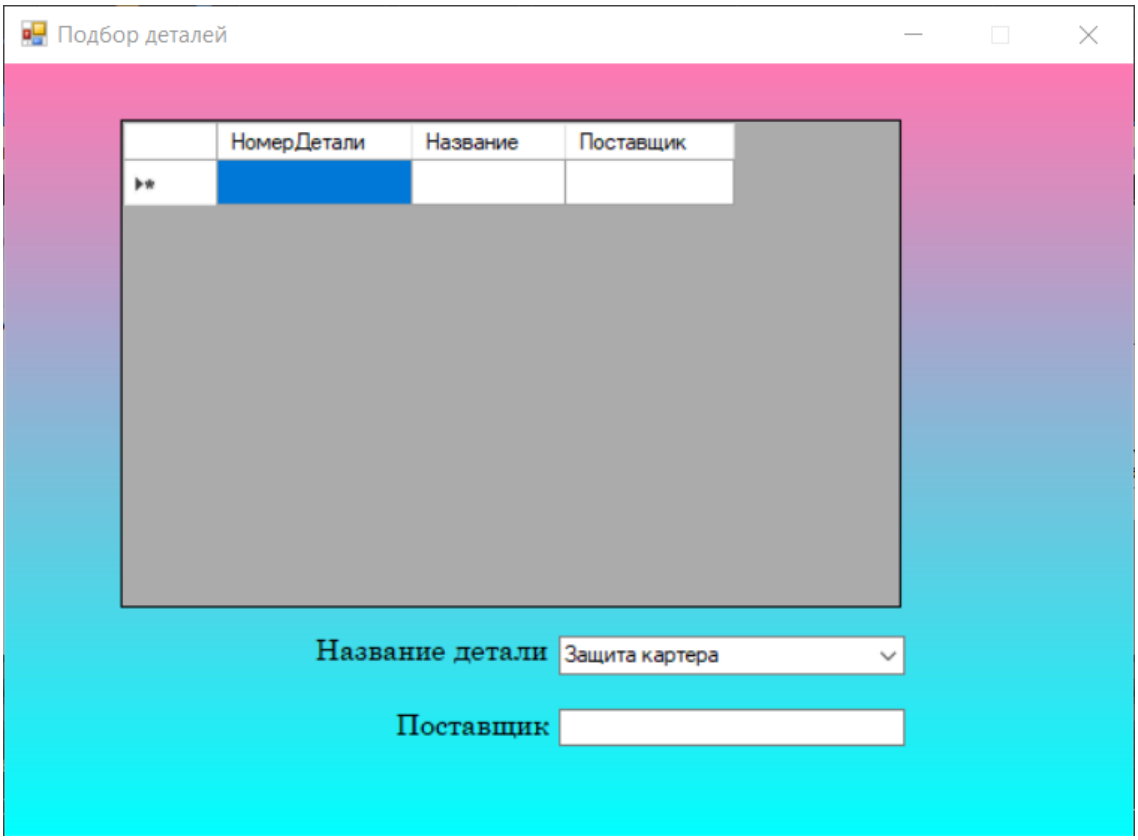


Рис. 2.10. Форма «Подбор деталей»

Форма создания клиента (Рис. 2.11., приложение В п.7) позволяет сотруднику добавлять новых клиентов в базу данных. Также реализован поиск по ФИО или по номеру телефона. Возможность редактирования табличной части заблокирована. При введение в поля создания клиента некорректных значений форма выведет ошибку с пояснением причины.

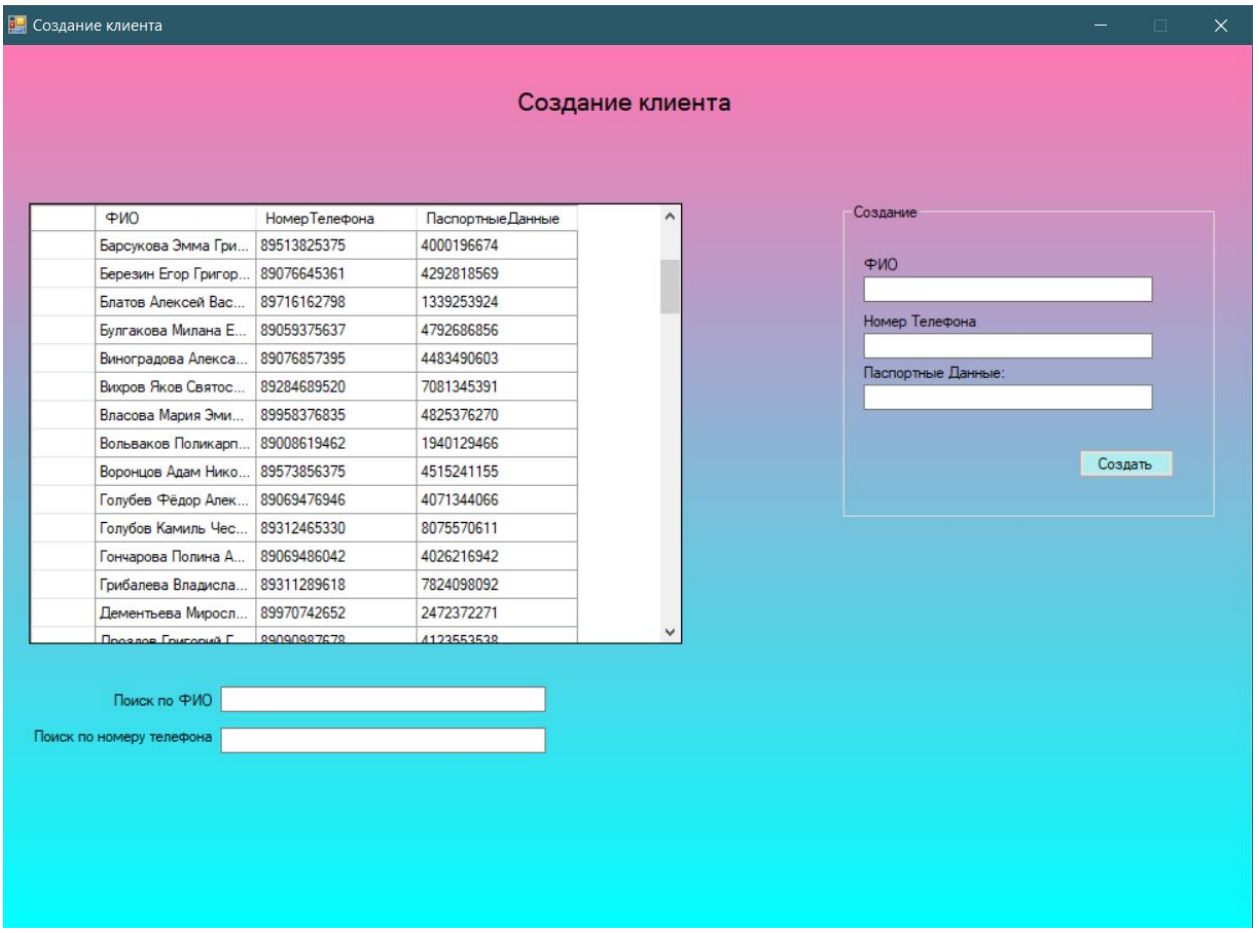


Рис. 2.11. Форма «Создание клиента»

На этом функции доступные консультантам заканчиваются.

Рассмотрим домашнюю страницу продавца (Рис. 2.12., приложение В п.8). Домашняя форма позволяет продавцу создавать сделки, удалять сделки, а также же редактировать записи о сделках. Работая на этой форме сотрудник может как добавлять так и удалять дополнительное оборудование в определенной сделке. На странице расположено 3 табличной части для удобства работы: одна выводит список всех сделок, вторая список всех деталей не участвующих в сделках, третья список деталей в выбранной сделке.

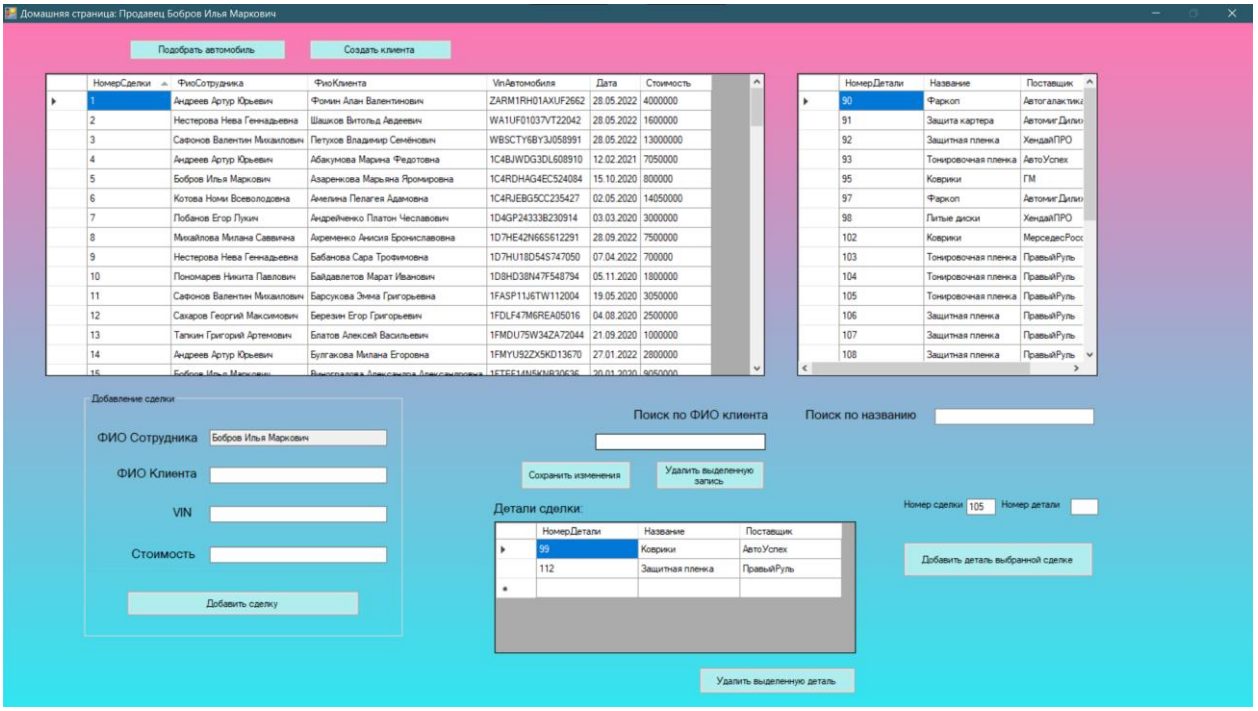


Рис. 2.12. Форма «Домашняя страница: Продавец»

На этом функции доступные продавцам заканчиваются.

Последняя роль доступная для выбора сотрудником – Механик. Домашняя форма механика (Рис. 2.13., приложение В п.9). позволяет сотруднику находить детали, которые необходимо установить на определенный автомобиль. Поиск возможно осуществлять как по номеру сделки, в которой участвует автомобиль, так и по VIN номеру автомобиля.

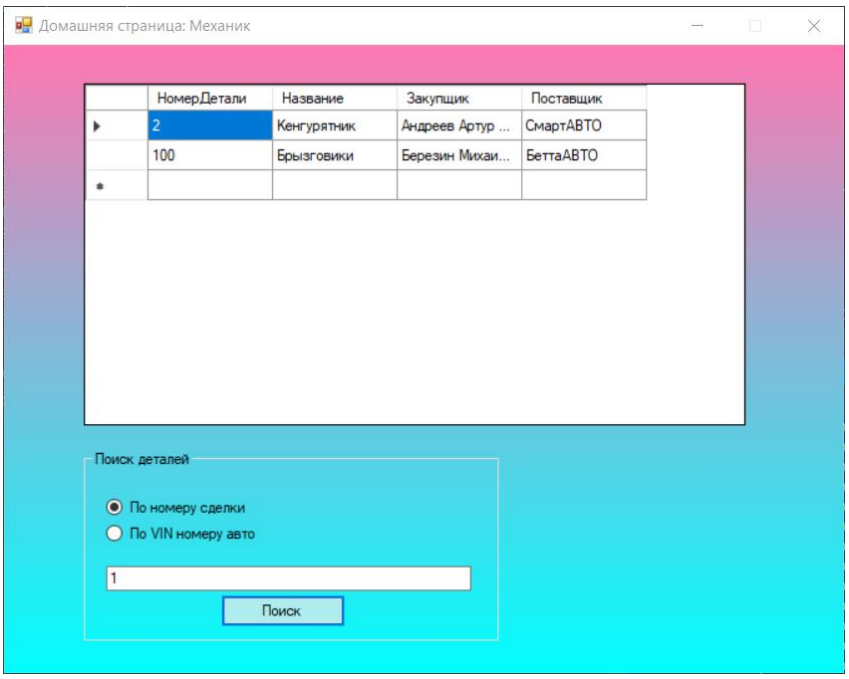


Рис. 2.13. Форма «Домашняя страница: Механик»

Заключение

В данном курсовом проекте рассматривалась деятельность автосалона. Автосалон – учреждение, имеющее большое количество сотрудников и задач, поэтому для более комфортной и продуктивной работы необходимо автоматизировать и оптимизировать основные процессы.

Актуальность данной работы заключается в необходимости хранения и обработки большого количества данных в электронном виде, позволяющий автоматизировать работу автосалона. Такого результата помогает добиться комплексная обработка и централизованное хранение данных. Благодаря этому, каждый сотрудник имеет доступ к данным и возможность работать с ними.

В ходе выполнения данного проекта были получены следующие результаты:

1. Анализ предметной области позволил определить актуальность темы. При проведении анализа функциональных возможностей уже существующей программы «1С Альфа-Авто» и сайта по продаже автомобилей Auto.ru были определены основные функции, которые должны быть реализованы в БД.

2. На этапе концептуального проектирования были выделены действующие лица (консультант, продавец, механик, кладовщик, главный менеджер) и их функции, на основе которых была построена диаграмма вариантов использования, с помощью которой были определены объекты, информацию о которых следует хранить в БД.

3. Основная задача этапа логического проектирования заключалась в построении и описании ER-диаграмм, на основе которых, при помощи правил, были построены отношения, определяющие реляционную схему базы данных автосалона: Автомобиль, Клиент, Сотрудник, Поставщик, Сделка, ДопОборудование, ДопОборудованиеСделки, ТестДрайв.

4. На заключительном этапе физического проектирования были сформулированы требования к структурам таблиц базы данных, реализованных с помощью MS SQL Server Management Studio и была создана схема БД

автосалона, отражающая связи между таблицами. Для обеспечения полноценного функционирования программный части были разработаны хранимые процедуры, пользовательские функции и триггеры.

5. Для ранее созданной базы данных был реализован пользовательский интерфейс с использованием языка C# и среды разработки MS Visual Studio.
На основе данных, которые были определены в процессе проектирования, была разработана БД, позволяющая выполнять различные функции в зависимости от роли пользователя. Для менеджера доступны следующие функции: изменение списка автомобилей, поставщиков, сотрудников, деталей, а также сбор информации о результативности продаж и остатках на складе. Для продавца – заключение сделок, подбор автомобиля, создание нового клиента. Для консультанта – подбор автомобилей, подбор деталей, запись на тест-драйв и создание нового клиента. Для механика доступен только поиск деталей по определенному автомобилю либо сделке.

В итоге был получен программный продукт полностью соответствующий поставленным задачам.

Список используемых источников

1. Сайт «Авто.ру» [Электронный ресурс] – Режим доступа: https://auto.ru/, свободный. Дата обращения 15.10.2021 г.
2. Сайт «1С:Альфа-Авто» [Электронный ресурс] – Режим доступа: https://1c-alfa-avto.ru/, свободный. Дата обращения 15.10.2021 г.
3. Сайт «АвтоДилер» [Электронный ресурс] – Режим доступа: https://autodealer.ru/, свободный. Дата обращения 15.10.2021 г.
4. Сайт «Хабр» [Электронный ресурс] – Режим доступа: https://habr.com/ru/post/440556/, свободный. Дата обращения 30.11.2021 г.
5. Сайт «Хабр» [Электронный ресурс] – Режим доступа: https://habr.com/ru/post/566218/, свободный. Дата обращения 10.12.2021 г.
6. Сайт «Документация по C#» [Электронный ресурс] – Режим доступа: https://learn.microsoft.com/ru-ru/dotnet/csharp/, свободный. Дата обращения 01.12.2022 г.

Приложение А

SQL-операторы создания процедур, функций и триггеров

1. Процедура «InsertCar»

```
USE [БД автосалона]
GO
/****** Object:  StoredProcedure [dbo].[InsertCar]    Script Date: 28.11.2022 17:45:56
******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
-- Author:           <Author,,Name>
-- Create date: <Create Date,,>
-- Description:      <Description,,>
-- =============================================
ALTER PROCEDURE [dbo].[InsertCar]
       @vin VARCHAR(20), @post varchar(50), @zak varchar(50),@mark varchar(50),@model
varchar(50),
       @year bigint,@type varchar(50), @color varchar(50), @mater varchar(50),@kpp
varchar(50), @engine varchar(50),
       @power INT,@price INT, @error INT OUTPUT
AS
BEGIN
SET @error=0
if @vin IN (SELECT vin FROM Автомобиль) SET @error=1
              ELSE
              IF (@year<=1950) SET @error =2;
                     ELSE
                     if (@power<=0) SET @error =3;
                            ELSE
                                   If (@price<=0) SET @error =4;
                                   ELSE
                                          INSERT INTO Автомобиль VALUES
(@vin,@post,@zak,@mark,@model,@year,@type,@color,@mater,@kpp,@engine,@power,@price)
END
```

2. Процедура «PriceByVin»

```
USE [БД автосалона]
GO
/****** Object:  StoredProcedure [dbo].[PriceByVIN]    Script Date: 28.11.2022 17:47:54
******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
-- Author:           <Author,,Name>
-- Create date: <Create Date,,>
-- Description:      <Description,,>
-- =============================================
ALTER PROCEDURE [dbo].[PriceByVIN]
       @vin varchar(17),@price money output
AS
BEGIN
       set @price=0
       Set @price=(select Стоимость from Автомобиль where VIN =@vin)
END
```

3. Табличная функция «CountOfCarsNotInDealByMark»

```
USE [БД автосалона]
GO
/****** Object:  UserDefinedFunction [dbo].[CountOfCarsNotInDealByMark]    Script Date:
28.11.2022 17:49:15 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
-- Author:           <Author,,Name>
-- Create date: <Create Date,,>
-- Description:      <Description,,>
-- =============================================
ALTER FUNCTION [dbo].[CountOfCarsNotInDealByMark]
(
)
RETURNS TABLE
AS
RETURN
(
       SELECT Марка, COUNT(*) AS ОставшеесяКоличество FROM Автомобиль
       WHERE VIN not in (SELECT VINАвтомобиля FROM Сделка)
       GROUP BY Марка
)
```

4. Табличная функция «CarsNotInDeal»

```
USE [БД автосалона]
GO
/****** Object:  UserDefinedFunction [dbo].[CarsNotInDeal]    Script Date: 28.11.2022
17:50:06 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
-- Author:           <Author,,Name>
-- Create date: <Create Date,,>
-- Description:      <Description,,>
-- =============================================
ALTER FUNCTION [dbo].[CarsNotInDeal]
(

)
RETURNS TABLE
AS
RETURN
(
       SELECT * FROM Автомобиль WHERE VIN NOT IN (SELECT VINАвтомобиля FROM Сделка)
)
```

5. Табличная функция «ModelsByMark»

```
USE [БД автосалона]
GO
/****** Object:  UserDefinedFunction [dbo].[ModelsByMark]    Script Date: 28.11.2022
17:50:58 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
-- Author:           <Author,,Name>
-- Create date: <Create Date,,>
-- Description:      <Description,,>
-- =============================================
```

```
ALTER FUNCTION [dbo].[ModelsByMark]
(
	@mark VARCHAR(20)

)
RETURNS TABLE
AS
RETURN
(
	SELECT DISTINCT Модель FROM Автомобиль WHERE Марка=@mark AND VIN NOT IN (SELECT
VINАвтомобиля FROM CarsInDeal())
)
```

6. Процедура «InsertDetail»

```
USE [БД автосалона]
GO
/****** Object:  StoredProcedure [dbo].[InsertDetail]    Script Date: 28.11.2022 17:51:45
******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
-- Author:		<Author,,Name>
-- Create date: <Create Date,,>
-- Description:	<Description,,>
-- =============================================
ALTER PROCEDURE [dbo].[InsertDetail]
	@id int, @name varchar(50), @zak varchar(50),@post varchar(50),
	@error int OUTPUT
AS
BEGIN
	IF @id NOT IN(SELECT НомерДетали FROM ДопОборудование)
		BEGIN
			INSERT INTO ДопОборудование VALUES (@id,@name,@zak,@post)
			SET @error=0
		END
	ELSE SET @error=1
END
```

7. Процедура «InsertPost»

```
USE [БД автосалона]
GO
/****** Object:  StoredProcedure [dbo].[InsertPost]    Script Date: 28.11.2022 17:52:32
******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
-- Author:		<Author,,Name>
-- Create date: <Create Date,,>
-- Description:	<Description,,>
-- =============================================
ALTER PROCEDURE [dbo].[InsertPost]
	@name varchar(50),@phone varchar(50),@error int output
AS
BEGIN
	set @error=0;
	if @name in (select Название From Поставщик) set @error=1;
	else
		if @phone in (select НомерТелефона From Поставщик) set @error=2;
		else
			insert into Поставщик values (@name,@phone)
```

```
END
```

8. Табличная функция «DetailsOfDeal»

```
USE [БД автосалона]
GO
/****** Object:  UserDefinedFunction [dbo].[DetailsOfDeal]    Script Date: 28.11.2022
17:53:33 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
-- Author:           <Author,,Name>
-- Create date: <Create Date,,>
-- Description:      <Description,,>
-- =============================================
ALTER FUNCTION [dbo].[DetailsOfDeal]
(
       @deal_num int
)
RETURNS TABLE
AS
RETURN
(
       SELECT ДопОборудование.НомерДетали, Название,Поставщик FROM ДопОборудование JOIN
ДопОборудованиеСделки
       ON ДопОборудование.НомерДетали=ДопОборудованиеСделки.НомерДетали
       WHERE ДопОборудованиеСделки.НомерСделки=@deal_num
)
```

9. Табличная функция «DetailNotInDeal»

```
USE [БД автосалона]
GO
/****** Object:  UserDefinedFunction [dbo].[DetailNotInDeal]    Script Date: 28.11.2022
17:54:13 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
-- Author:           <Author,,Name>
-- Create date: <Create Date,,>
-- Description:      <Description,,>
-- =============================================
ALTER FUNCTION [dbo].[DetailNotInDeal]
(
)
RETURNS TABLE
AS
RETURN
(
       SELECT * FROM ДопОборудование WHERE НомерДетали NOT IN (SELECT НомерДетали FROM
ДопОборудованиеСделки)
)
```

10. Табличная функция «CountOfDetailsNotInDealByName»

```
USE [БД автосалона]
GO
/****** Object:  UserDefinedFunction [dbo].[CountOfDetailsNotInDealByName]    Script Date:
28.11.2022 17:55:37 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
```

```
-- =============================================
-- Author:		<Author,,Name>
-- Create date: <Create Date,,>
-- Description:	<Description,,>
-- =============================================
ALTER FUNCTION [dbo].[CountOfDetailsNotInDealByName]
(

)
RETURNS TABLE
AS
RETURN
(
	SELECT Название, COUNT(*) AS ОстаткиДеталей FRom ДопОборудование
	WHERE НомерДетали NOT IN (SELECT НомерДетали FROm ДопОборудованиеСделки)
	group by Название
)
```

11. Процедура «InsertSotr»

```
USE [БД автосалона]
GO
/****** Object:  StoredProcedure [dbo].[InsertSotr]    Script Date: 28.11.2022 17:56:17
******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
-- Author:		<Author,,Name>
-- Create date: <Create Date,,>
-- Description:	<Description,,>
-- =============================================
ALTER PROCEDURE [dbo].[InsertSotr]
@fio varchar(50),@phone varchar(50),@dol varchar(50),@error INT OUTPUT
AS
BEGIN
Set @error=0;
if @fio in (select ФИО from Сотрудник)
	set @error=1
	else
		If @phone in (select НомерТелефона from Сотрудник)
			set @error=2
		else
			INSERT INTO Сотрудник VALUES (@fio,@phone,@dol);
END
```

12. Табличная функция «ManagersList»

```
USE [БД автосалона]
GO
/****** Object:  UserDefinedFunction [dbo].[Managerslist]    Script Date: 28.11.2022
17:57:05 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
-- Author:		<Author,,Name>
-- Create date: <Create Date,,>
-- Description:	<Description,,>
-- =============================================
ALTER FUNCTION [dbo].[Managerslist]
(
)
RETURNS TABLE
```

```
AS
RETURN
(
	SELECT ФИО FROM Сотрудник WHERE Должность='Менеджер'
)
```

13. Табличная функция «ManagersAndStorekeepers»

```
USE [БД автосалона]
GO
/****** Object:  UserDefinedFunction [dbo].[ManagersAndStorekeepers]    Script Date:
28.11.2022 17:57:40 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
-- Author:		<Author,,Name>
-- Create date: <Create Date,,>
-- Description:	<Description,,>
-- =============================================
ALTER FUNCTION [dbo].[ManagersAndStorekeepers]
()
RETURNS TABLE
AS
RETURN
(
	SELECT ФИО FROM Сотрудник WHERE Должность='Менеджер' OR Должность='Кладовщик'
)
```

14. Процедура «InsertDealDetail»

```
USE [БД автосалона]
GO
/****** Object:  StoredProcedure [dbo].[InsertDealDetail]    Script Date: 28.11.2022
17:58:26 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
-- Author:		<Author,,Name>
-- Create date: <Create Date,,>
-- Description:	<Description,,>
-- =============================================
ALTER PROCEDURE [dbo].[InsertDealDetail]
@deal int,@detail int
as
BEGIN
	INSErt into ДопОборудованиеСделки VALUES (@deal,@detail)
END
```

15. Процедура «InsertClient»

```
USE [БД автосалона]
GO
/****** Object:  StoredProcedure [dbo].[InsertClient]    Script Date: 28.11.2022 17:59:20
******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
-- Author:		<Author,,Name>
-- Create date: <Create Date,,>
```

```
-- Description:      <Description,,>
-- ============================================
ALTER PROCEDURE [dbo].[InsertClient]
       @fio varchar(50), @phone varchar(50),@pasp varchar(50),@error INT output
AS
BEGIN
set @error=0
       if @fio IN (SELECT ФИО FROM Клиент) SET @error=1
       else
              if @phone IN (SELECT НомерТелефона FROM Клиент) SET @error=2
              else
                     if @pasp IN (SELECT ПаспортныеДанные FROM Клиент) SET @error=3
                     else
                            INSERT INTO Клиент values (@fio,@phone,@pasp)
END
```

16. Процедура «InsertTestDrive»

```
USE [БД автосалона]
GO
/****** Object:  StoredProcedure [dbo].[InsertTestDrive]    Script Date: 28.11.2022
18:00:02 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- ============================================
-- Author:           <Author,,Name>
-- Create date: <Create Date,,>
-- Description:      <Description,,>
-- ============================================
ALTER PROCEDURE [dbo].[InsertTestDrive]
       @date date, @time time, @vin VARCHAR(17), @fio VARCHAR(50),
       @error INT Output
       AS
BEGIN
SET @error=0
DECLARE @num INT
SET @num=(SELECT max(НомерТестДрайва) FROM ТестДрайв)
SET @num =@num+1
IF @fio NOT IN(SELECT ФИО FROM Клиент)
       SET @error=1
       ELSE
              IF @vin Not in (select Vin from CarsNotInDeal())
                     SET @error=2
                     ELSE
                            IF @date IN (SELECT Дата FROM ТестДрайв) AND EXISTS(SELECT *
FROM ТестДрайв WHERE Время BETWEEN dateadd(hour,-1,@time) AND dateadd(hour,1,@time) AND
VINАвтомобиля=@vin AND Дата=@date)
                            SET @error=3
                            ELSE
                            INSERT INTO ТестДрайв VALUES (@num,@date,@time,@vin,@fio)
END
```

17. Процедура «InsertDeal»

```
USE [БД автосалона]
GO
/****** Object:  StoredProcedure [dbo].[InsertDeal]    Script Date: 28.11.2022 18:00:46
******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
```

```
-- =============================================
-- Author:           <Author,,Name>
-- Create date: <Create Date,,>
-- Description:      <Description,,>
-- =============================================
ALTER PROCEDURE [dbo].[InsertDeal]
       @sotr varchar(50), @client varchar(50),@vin varchar(17),@date date,
       @price money,@error INT output
AS
BEGIN
DECLARE @indx int
set @indx=(SELECT max(НомерСделки) FROM Сделка)
set @indx=@indx+1;
set @error=0;
if @vin in (Select * from CarsInDeal())
              set @error=1
       else
              if @client NOT IN (select ФИО FRom Клиент)
                     set @error=3
                     else
                            if @price< (select Стоимость FROM Автомобиль WHERE VIN=@vin)
                                   set @error=2
                            else
                                   INSERT INTO Сделка values
(@indx,@sotr,@client,@vin,@date,@price)


END
```

18. Табличная функция «SellsByMonth»

```
USE [БД автосалона]
GO
/****** Object:  UserDefinedFunction [dbo].[SellsByMonth]    Script Date: 28.11.2022
18:02:12 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
-- Author:           <Author,,Name>
-- Create date: <Create Date,,>
-- Description:      <Description,,>
-- =============================================
ALTER FUNCTION [dbo].[SellsByMonth]
(
       @date date
)
RETURNS TABLE
AS
RETURN
(
       SELECT ФИО,SUM(Стоимость) AS ЗаработанноеВМесяце from Сотрудник left JOIN Сделка
       on Сотрудник.ФИО=Сделка.ФиоСотрудника
       where month(Сделка.Дата)=month(@date) AND year(Сделка.Дата)=year(@date)
       group by ФИО
)
```

19. Триггер «DeleteDetailInDeal»

```
USE [БД автосалона]
GO
/****** Object:  Trigger [dbo].[DeleteDetailsInDeal]    Script Date: 28.11.2022 18:02:48
******/
SET ANSI_NULLS ON
```

```
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER trigger [dbo].[DeleteDetailsInDeal]
on [dbo].[Сделка]
INSTEAD OF DELETE
as
begin
declare @deal_num INT
SET @deal_num=(SELECT НомерСделки FROM deleted)
if exists(SELECT * from ДопОборудованиеСделки WHERE НомерСделки=@deal_num)
       DELETe from ДопОборудованиеСделки WHERE НомерСделки=@deal_num
DELETE from Сделка WHERE НомерСделки=@deal_num
end
```

Приложение Б

Заполнение таблиц данными

VIN	Поставщик	Закупщик	Марка	Модель	ГодВыпуска	ТипКузова	Цвет	Материал...	КоробкаП...	ТипДвигат...	Мощность	Стоимость
VSA2043ZN...	МерседесР...	Котова Но...	Mercedes-B...	S550	2003	Седан	Черный	Кожанный	Автоматич...	Бензиновый	500	3000000,00...
WA1UF0103...	АвтоРесурс	Андреев А...	Audi	A1	2020	Хэтчбек	Черный	Тряпичный	Механичес...	Бензиновый	180	15000000,0...
WBSCTY6BY...	СмартАВТО	Котова Но...	BMW	iX	2022	Кроссовер	Серый	Кожанный	Автоматич...	Электричес...	619	12800000,0...
ZARM1RH0...	СмартАВТО	Сафонов В...	Alfa Romeo	Julietta	2016	Купе	Красный	Кожанный	Автоматич...	Бензиновый	250	3800000,00...

Рис.Б.1. Данные таблицы «Автомобиль»

НомерДетали	Название	Закупщик	Поставщик
1	Рейлинг	Андреев А...	СмартАВТО
2	Кенгурятник	Андреев А...	СмартАВТО
3	Защита картера	Котова Но...	АвтоРесурс
4	Защитная пленка	Сафонов В...	СмартАВТО

Рис.Б.2. Данные таблицы «ДопОборудование»

НомерСделки	НомерДетали
1	2
2	3
3	1

Рис.Б.3. Данные таблицы «ДопОборудовниеСделки»

ФИО	НомерТелефона	ПаспортныеДанные
Петухов Влади...	89560989065	5468398603
Фомин Алан В...	89876759876	8674758379
Шашков Витол...	89069605423	7806849682

Рис.Б.4. Данные таблицы «Клиент»

Название	НомерТелефона
АвтоРесурс	89344354365
СмартАВТО	89430935423
МерседесРоссия	89543953922

Рис.Б.5. Данные таблицы «Поставщик»

НомерСделки	ФиоСотрудника	ФиоКлиента	VinАвтомо...	Дата	Стоимость
1	Андреев Артур...	Фомин Алан ...	ZARM1RH0...	2022-05-28	4000000,00...
2	Нестерова Нев...	Шашков Вито...	WA1UF0103...	2022-05-28	1600000,00...
3	Сафонов Вале...	Петухов Влад...	WBSCTY6BY...	2022-05-28	13000000,0...

Рис.Б.6. Данные таблицы «Сделка»

ФИО	НомерТелефо...	Должность
Андреев Арту...	89530929502	Продавец
Котова Номи ...	89405538336	Продавец
Нестерова Не...	89463588543	Менеджер
Сафонов Вал...	89049043535	Продавец

Рис.Б.7. Данные таблицы «Сотрудник»

НомерТестДрайва	Дата	Время	VINАвтомобиля	ФиоКлиента
1	2022-05-28	10:00:00	WBSCTY6BY3J05...	Фомин Алан В...
2	2022-05-28	09:00:00	VSA2043ZN21F5...	Шашков Витол...
3	2022-05-28	10:30:00	ZARM1RH01AXU...	Петухов Влади...

Рис.Б.8. Данные таблицы «ТестДрайв»

Приложение В

Код программы

1. Форма «Селектор должности»

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace CarsDB
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"бД_автосалонаDataSet.Сотрудник". При необходимости она может быть перемещена или удалена.
            this.сотрудникTableAdapter.Fill(this.бД_автосалонаDataSet.Сотрудник);
            this.сотрудникBindingSource.Filter = "Должность='Продавец'";
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"бД_автосалонаDataSet.Managerslist". При необходимости она может быть перемещена или
удалена.
            this.managerslistTableAdapter.Fill(this.бД_автосалонаDataSet.Managerslist);

        }

        private void label1_Click(object sender, EventArgs e)
        {

        }

        private void button1_Click(object sender, EventArgs e)
        {

        }

        private void label3_Click(object sender, EventArgs e)
        {

        }
        private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
        {
            if (comboBox1.Text == "Менеджер")
            {
                managerslistComboBox.Visible = true;
            }
            else managerslistComboBox.Visible = false;
            if (comboBox1.Text == "Продавец")
            {
                salesmanComboBox.Visible = true;
            }
            else salesmanComboBox.Visible = false;
```

```
        }
        private void button1_Click_1(object sender, EventArgs e)
        {
            string val = comboBox1.Text;
            switch (val)
            {
                case "":
                    Console.WriteLine("Введите роль");
                    break;
                case "Менеджер":
                    this.Hide();
                    Form ManagerHome = new ManagerHome(managerslistComboBox.Text);
                    ManagerHome.Show();
                    break;
                case "Продавец":
                    this.Hide();
                    Form SalesmanHome = new SalesmanHome(salesmanComboBox.Text);
                    SalesmanHome.Show();
                    break;
                case "Консультант":
                    this.Hide();
                    Form ConsultantHome = new ConsultantHome();
                    ConsultantHome.Show();
                    break;
                case "Механик":
                    this.Hide();
                    Form MechanicEquipment = new MechanicEquipment();
                    MechanicEquipment.Show();
                    break;
                case "Кладовщик":
                    this.Hide();
                    Form StorekeeperHome = new StorekeeperHome();
                    StorekeeperHome.Show();
                    break;
            }
        }
    }
}
```

2. Форма «Домашняя страница: Менеджер»

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace CarsDB
{
    public partial class ManagerHome : Form
    {
        string position;
        public ManagerHome(string pos)
        {
            InitializeComponent();
            this.position = pos;
            this.Text = this.Text + " " + this.position;
            zaktextBox.Text = this.position;
        }
        public void ManagerHome_FormClosed(object sender, FormClosedEventArgs e)
        {
```

```
            Form frm = Application.OpenForms[0];
            frm.Show();
        }
        private void button1_Click(object sender, EventArgs e)
        {
            Form frm = new CreateDetail(this.position);
            frm.Show();
        }
        private void ClearInsert(object sender, EventArgs e)
        {
            vintextBox.Text = "";
            marktextBox.Text = "";
            modeltextBox.Text = "";
            yeartextBox.Text = "";
            typecomboBox.SelectedIndex = 0;
            colorcomboBox.SelectedIndex = 0;
            matherialcomboBox.SelectedIndex = 0;
            kppcomboBox.SelectedIndex = 0;
            enginecomboBox.SelectedIndex = 0;
            powertextBox.Text = "";
            pricetextBox.Text = "";

        }

        private void tabPage1_Click(object sender, EventArgs e)
        {

        }

        private void label1_Click(object sender, EventArgs e)
        {

        }

        private void dateTimePicker1_ValueChanged(object sender, EventArgs e)
        {

        }

        private void ManagerHome_Load(object sender, EventArgs e)
        {
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"6Д_автосалонаDataSet.CountOfDetailsNotInDealByName". При необходимости она может быть
перемещена или удалена.

this.countOfDetailsNotInDealByNameTableAdapter.Fill(this.6Д_автосалонаDataSet.CountOfDetail
sNotInDealByName);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"6Д_автосалонаDataSet.CountOfCarsNotInDealByMark". При необходимости она может быть
перемещена или удалена.

this.countOfCarsNotInDealByMarkTableAdapter.Fill(this.6Д_автосалонаDataSet.CountOfCarsNotIn
DealByMark);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"6Д_автосалонаDataSet.Сотрудник". При необходимости она может быть перемещена или удалена.
            this.сотрудникTableAdapter.Fill(this.6Д_автосалонаDataSet.Сотрудник);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"6Д_автосалонаDataSet.Managerslist". При необходимости она может быть перемещена или
удалена.
            this.managerslistTableAdapter.Fill(this.6Д_автосалонаDataSet.Managerslist);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"6Д_автосалонаDataSet.Поставщик". При необходимости она может быть перемещена или удалена.
            this.поставщикTableAdapter.Fill(this.6Д_автосалонаDataSet.Поставщик);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"6Д_автосалонаDataSet.Автомобиль". При необходимости она может быть перемещена или удалена.
```

```
            this.автомобильTableAdapter.Fill(this.бД_автосалонаDataSet.Автомобиль);
            this.sellsByMonthTableAdapter.Fill(this.бД_автосалонаDataSet.SellsByMonth,
dateTimePicker1.Text);

this.countOfDetailsNotInDealByNameTableAdapter.Fill(this.бД_автосалонаDataSet.CountOfDetail
sNotInDealByName);

this.countOfCarsNotInDealByMarkTableAdapter.Fill(this.бД_автосалонаDataSet.CountOfCarsNotIn
DealByMark);

        }

        private void dataGridView1_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {

        }

        private void button2_Click(object sender, EventArgs e)
        {
            if (vintextBox.Text.Length < 17) { MessageBox.Show("Некорректный vin"); return;
}
            int? error = 0;
            автомобильTableAdapter.InsertCar(vintextBox.Text, postcomboBox.Text,
this.position, marktextBox.Text, modeltextBox.Text,
Convert.ToInt64(yeartextBox.Text),typecomboBox.Text, colorcomboBox.Text,
matherialcomboBox.Text, kppcomboBox.Text, enginecomboBox.Text,
Convert.ToInt32(powertextBox.Text), Convert.ToInt32(pricetextBox.Text), ref error);
            if (error == 0)
            {
                MessageBox.Show("Выполнено успешно","Добавление автомобиля");
                автомобильTableAdapter.Fill(this.бД_автосалонаDataSet.Автомобиль);
                if (!radioButton1.Checked) ClearInsert(sender,e);
                UpdateOst(sender, e);
            }
            if (error==1)
                MessageBox.Show("Повторяющийся VIN");
            if (error == 2)
                MessageBox.Show("Некорректный год выпуска");
            if (error == 3)
                MessageBox.Show("Нулевая мощность");
            if (error == 4)
                MessageBox.Show("Нулевая цена");
        }

        private void button3_Click(object sender, EventArgs e)
        {
            if (searchtextBox.Text=="")
            {
                автомобильBindingSource.Filter = "";
                return;
            }
            if (searchtextBox.Text.Length < 17) { MessageBox.Show("Некорректный vin");
return; }
            автомобильBindingSource.Filter = "VIN='" +
Convert.ToString(searchtextBox.Text)+"'";
        }

        private void button4_Click(object sender, EventArgs e)
        {
            try
            {
                автомобильBindingSource.RemoveCurrent();
                this.Validate();
                this.автомобильBindingSource.EndEdit();
```

```
                автомобильTableAdapter.Update(бД_автосалонаDataSet);
                UpdateOst(sender, e);
            }
            catch
            {
                this.автомобильTableAdapter.Fill(this.бД_автосалонаDataSet.Автомобиль);
                MessageBox.Show("Попытка удаления связанного значения", "Ошибка");
            }
        }

        private void button5_Click(object sender, EventArgs e)
        {
            this.Validate();
            this.автомобильBindingSource.EndEdit();
            try
            {
                автомобильTableAdapter.Update(бД_автосалонаDataSet);
            }
            catch (Exception ex)
            {
                MessageBox.Show("Обновление значений невозможно", "ERROOR");
                Console.WriteLine(ex);
                this.автомобильTableAdapter.Fill(this.бД_автосалонаDataSet.Автомобиль);
            }
        }

        private void textBox1_TextChanged(object sender, EventArgs e)
        {
            поставщикBindingSource1.Filter = "Названиe LIKE'" + searchnametextBox.Text +
"%'";
            поставщикTableAdapter.Update(бД_автосалонаDataSet);
        }

        private void button7_Click(object sender, EventArgs e)
        {
            this.Validate();
            this.поставщикBindingSource1.EndEdit();
            try
            {
                поставщикTableAdapter.Update(бД_автосалонаDataSet);
            }
            catch (Exception ex)
            {
                MessageBox.Show("Обновление значений невозможно", "Изменение поставщика");
                Console.WriteLine(ex);
                this.поставщикTableAdapter.Fill(this.бД_автосалонаDataSet.Поставщик);
            }
        }

        private void button6_Click(object sender, EventArgs e)
        {
            try
            {
                поставщикBindingSource1.RemoveCurrent();
                this.Validate();
                this.поставщикBindingSource1.EndEdit();
                поставщикTableAdapter.Update(бД_автосалонаDataSet);
            }
            catch
            {
                this.автомобильTableAdapter.Fill(this.бД_автосалонаDataSet.Автомобиль);
                MessageBox.Show("Попытка удаления связанного значения", "Удаление
поставщика");
            }
        }
```

```
        private void label16_Click(object sender, EventArgs e)
        {

        }

        private void label17_Click(object sender, EventArgs e)
        {

        }

        private void textBox2_TextChanged(object sender, EventArgs e)
        {
            sotrphoneerrorProvider.Clear();
        }

        private void textBox1_TextChanged_1(object sender, EventArgs e)
        {
            sotrfioerrorProvider.Clear();
        }

        private void textBox3_TextChanged(object sender, EventArgs e)
        {

        }

        private void label15_Click(object sender, EventArgs e)
        {

        }

        private void button8_Click(object sender, EventArgs e)
        {
            if (phonetextBox.TextLength<11)
            {
                MessageBox.Show("Введите корректный номер телефона", "Добавление
сотрудника");
                return;
            }
            int? error = 0;
            сотрудникTableAdapter.InsertSotr(fiotextBox.Text,
phonetextBox.Text,dolcomboBox.Text, ref error);
            if (error == 1)
            {
                sotrfioerrorProvider.SetError(fiotextBox, "Сотрудник с таким ФИО уже
существует");
                return;
            }
            if (error ==2)
            {
                sotrphoneerrorProvider.SetError(phonetextBox, "Сотрудник с таким номером
телефона уже существует");
                return;
            }
            else
            {
                MessageBox.Show("Выполнено успешно", "Добавление сотрудника");
                сотрудникTableAdapter.Fill(бД_автосалонаDataSet.Сотрудник);
            }

        }

        private void textBox1_TextChanged_2(object sender, EventArgs e)
        {
```

```
            if (((сотрудникBindingSource.Filter != "") && (comboBox1.Text == ""))||
(сотрудникBindingSource.Filter == ""))
            {
                сотрудникBindingSource.Filter = "ФИО LIKE'" + searchfiotextBox.Text + "%'";
                сотрудникTableAdapter.Update(бД_автосалонаDataSet);
            }
            else
            {
                if (searchfiotextBox.Text == "")
                {
                    сотрудникBindingSource.Filter = "Должность ='" + comboBox1.Text + "'";
                    сотрудникTableAdapter.Update(бД_автосалонаDataSet);
                }
                else
                {
                    сотрудникBindingSource.Filter = сотрудникBindingSource.Filter + "AND
ФИО LIKE'" + searchfiotextBox.Text + "%'";
                    сотрудникTableAdapter.Update(бД_автосалонаDataSet);
                }
            }
        }

        private void button9_Click(object sender, EventArgs e)
        {
            try
            {
                сотрудникBindingSource.RemoveCurrent();
                this.Validate();
                this.сотрудникBindingSource.EndEdit();
                сотрудникTableAdapter.Update(бД_автосалонаDataSet);
            }
            catch
            {
                this.сотрудникTableAdapter.Fill(this.бД_автосалонаDataSet.Сотрудник);
                MessageBox.Show("Попытка удаления связанного значения", "Удаление
сотрудника");
            }
        }

        private void button10_Click(object sender, EventArgs e)
        {
            this.Validate();
            this.сотрудникBindingSource.EndEdit();
            try
            {
                сотрудникTableAdapter.Update(бД_автосалонаDataSet);
            }
            catch (Exception ex)
            {
                MessageBox.Show("Обновление значений невозможно", "Изменение сотрудника");
                Console.WriteLine(ex);
                this.сотрудникTableAdapter.Fill(this.бД_автосалонаDataSet.Сотрудник);
            }
        }

        private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
        {
            if (comboBox1.Text != "")
            {
                if (searchfiotextBox.Text == "")
                {
                    сотрудникBindingSource.Filter = "Должность ='" + comboBox1.Text + "'";
                    сотрудникTableAdapter.Update(бД_автосалонаDataSet);
                }
                else
```

```
                {
                    сотрудникBindingSource.Filter = сотрудникBindingSource.Filter+ "AND
Должность ='" + comboBox1.Text + "'";
                    сотрудникTableAdapter.Update(бД_автосалонаDataSet);
                }
            }
            else
                if (searchfiotextBox.Text !="")
                {
                    сотрудникBindingSource.Filter = "ФИО LIKE'" + searchfiotextBox.Text +
"%'";
                    сотрудникTableAdapter.Update(бД_автосалонаDataSet);
                }
                else
                {
                    сотрудникBindingSource.Filter = "";
                    сотрудникTableAdapter.Update(бД_автосалонаDataSet);
                }
        }

        private void fillToolStripButton_Click(object sender, EventArgs e)
        {
            try
            {

//this.sellsByMonthTableAdapter.Fill(this.бД_автосалонаDataSet.SellsByMonth,
dateToolStripTextBox.Text);
            }
            catch (System.Exception ex)
            {
                System.Windows.Forms.MessageBox.Show(ex.Message);
            }

        }

        private void dateTimePicker1_ValueChanged_1(object sender, EventArgs e)
        {
            this.sellsByMonthTableAdapter.Fill(this.бД_автосалонаDataSet.SellsByMonth,
dateTimePicker1.Text);
        }

        private void tabPage3_Click(object sender, EventArgs e)
        {

this.countOfDetailsNotInDealByNameTableAdapter.Fill(this.бД_автосалонаDataSet.CountOfDetail
sNotInDealByName);

this.countOfCarsNotInDealByMarkTableAdapter.Fill(this.бД_автосалонаDataSet.CountOfCarsNotIn
DealByMark);
        }
        private void UpdateOst(object sender, EventArgs e)
        {

this.countOfDetailsNotInDealByNameTableAdapter.Fill(this.бД_автосалонаDataSet.CountOfDetail
sNotInDealByName);

this.countOfCarsNotInDealByMarkTableAdapter.Fill(this.бД_автосалонаDataSet.CountOfCarsNotIn
DealByMark);
        }

        private void button11_Click(object sender, EventArgs e)
        {
            if (postnametextBox.Text == "")
            {
                postnameerrorProvider.SetError(postnametextBox,"Введите название");
```

```
                return;

            }
            if (postphonetextBox.Text == "")
            {
                postphoneerrorProvider.SetError(postphonetextBox, "Введите номер
телефона");
                return;

            }
            int? error = 0;
            поставщикTableAdapter.InsertPost(postnametextBox.Text, postphonetextBox.Text,
ref error);
            if(error==0)
            {
                MessageBox.Show("Выполнено успешно", "Добавление поставщика");
                this.поставщикTableAdapter.Fill(this.бД_автосалонаDataSet.Поставщик);
            }
            if(error==1)
            {
                postnameerrorProvider.SetError(postnametextBox, "Поставщик с таким
названием уже существует");
            }
            if (error == 2)
            {
                postphoneerrorProvider.SetError(postphonetextBox, "Поставщик с таким
номером телефона уже существует");
            }




        }

        private void postnametextBox_TextChanged(object sender, EventArgs e)
        {
            postnameerrorProvider.Clear();
        }

        private void postphonetextBox_TextChanged(object sender, EventArgs e)
        {
            postphoneerrorProvider.Clear();
        }

        private void countOfDetailsNotInDealByNameDataGridView_CellContentClick(object
sender, DataGridViewCellEventArgs e)
        {

        }
    }
}
```

3. Форма «Домашняя страница: Консультант»

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
```

```
namespace CarsDB
{
    public partial class ConsultantHome : Form
    {
        public ConsultantHome()
        {
            InitializeComponent();
        }
        public void ConsultantHome_FormClosed(object sender, FormClosedEventArgs e)
        {
            Form frm = Application.OpenForms[0];
            frm.Show();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            Form ConsultantFindCar = new ConsultantFindCar();
            ConsultantFindCar.Show();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            Form TestDriveForm = new TestDriveForm();
            TestDriveForm.Show();
        }

        private void button3_Click(object sender, EventArgs e)
        {
            Form CreateClient = new CreateClient();
            CreateClient.Show();
        }

        private void ConsultantHome_Load(object sender, EventArgs e)
        {

        }

        private void gradientPanel2_Paint(object sender, PaintEventArgs e)
        {

        }

        private void button4_Click(object sender, EventArgs e)
        {
            Form frm = new FindDetail();
            frm.Show();
        }
    }
}
```

4. Форма «Редактирование списка тест-драйвов»

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
```

```
namespace CarsDB
{
    public partial class TestDriveForm : Form
    {
        public TestDriveForm()
        {
            InitializeComponent();
            datedateTimePicker1.MinDate = DateTime.Today;
            this.тестДрайвTableAdapter.Fill(this.бД_автосалонаDataSet.ТестДрайв);
            тестДрайвBindingSource.Filter = "Дата ='" + datedateTimePicker1.Value.Date +
"'";
        }
        public void TestDriveForm_FormClosed(object sender, FormClosedEventArgs e)
        {

        }
        public void UpdateTable(object sender, EventArgs e)
        {
            if (checkBox1.Checked)
            {
                тестДрайвBindingSource.Filter = "ФиоКлиента LIKE'" + searchfiotextBox.Text
+ "%'";

            }
            else
            {
                тестДрайвBindingSource.Filter = "ФиоКлиента LIKE'" + searchfiotextBox.Text
+ "%' AND Дата='" + datedateTimePicker1.Value.Date + "'";

            }

        }

        private void тестДрайвBindingNavigatorSaveItem_Click(object sender, EventArgs e)
        {
            this.Validate();
            this.тестДрайвBindingSource.EndEdit();
            this.tableAdapterManager.UpdateAll(this.бД_автосалонаDataSet);

        }

        private void TestDriveForm_Load(object sender, EventArgs e)
        {

        }

        private void button1_Click(object sender, EventArgs e)
        {
            Form CreateClient = new CreateClient();
            CreateClient.Show();
        }

        private void времяTextBox_TextChanged(object sender, EventArgs e)
        {

        }

        private void времяLabel_Click(object sender, EventArgs e)
        {

        }

        private void времяDateTimePicker_ValueChanged(object sender, EventArgs e)
        {
```

```
        }

        private void времяLabel1_Click(object sender, EventArgs e)
        {

        }

        private void button3_Click(object sender, EventArgs e)
        {
            if (datedateTimePicker1.Text != "" && timedateTimePicker2.Text != "" &&
vintextBox1.Text != "" && fiotextBox1.Text != "")
            {
                int? error = 0;
                string stime = timedateTimePicker2.Value.TimeOfDay.ToString();
                Console.WriteLine(stime);
                stime = stime.Substring(stime.Length-2, 3);
                Console.WriteLine(stime);
                DateTime time = Convert.ToDateTime(stime);
                Console.WriteLine(stime);
                тестДрайвTableAdapter.InsertTestDrive(datedateTimePicker1.Value,
time.TimeOfDay, vintextBox1.Text, fiotextBox1.Text,ref error);
                if (error == 0) { MessageBox.Show("Успешно добавлено","Добавление записи");
return; }
                if (error == 1) {fioerrorProvider.SetError(fiotextBox1, "Такого клиента не
существует"); return; }
                if (error == 2) {vinerrorProvider.SetError(vintextBox1, "Автомобиль с таким
VIN недоступен"); return;}
                if (error == 3) {timeerrorProvider.SetError(timedateTimePicker2, "Это время
занято");return; }
}
            else
            {
               if (datedateTimePicker1.Text == "")
{dateerrorProvider.SetError(datedateTimePicker1, "Выберите дату"); return; }
                if (timedateTimePicker2.Text == "")
{timeerrorProvider.SetError(timedateTimePicker2, "Выберите время"); return; }
                if (vintextBox1.TextLength< 17) {vinerrorProvider.SetError(vintextBox1,
"Введите корректный VIN"); return; }
                if (fiotextBox1.Text == "") {fioerrorProvider.SetError(fiotextBox1,
"Введите ФИО"); return; }
            }
            UpdateTable(sender, e);
            тестДрайвTableAdapter.Fill(this.бД_автосалонаDataSet.ТестДрайв);
        }

        private void dateDateTimePicker_ValueChanged(object sender, EventArgs e)
        {
            Console.WriteLine(datedateTimePicker1.Value.Date);
            UpdateTable(sender, e);
        }

        private void vinTextBox_TextChanged(object sender, EventArgs e)
        {
            vinerrorProvider.Clear();
        }

        private void fioTextBox_TextChanged(object sender, EventArgs e)
        {
            fioerrorProvider.Clear();
        }

        private void dataGridView1_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {
```

```
}

private void button2_Click(object sender, EventArgs e)
{
    Form frm = new ConsultantFindCar();
    frm.Show();
}

private void searchfiotextBox_TextChanged(object sender, EventArgs e)
{
    UpdateTable(sender, e);

}

private void button4_Click(object sender, EventArgs e)
{

}

private void button3_Click_1(object sender, EventArgs e)
{
    тестДрайвBindingSource.RemoveCurrent();
    this.Validate();
    this.тестДрайвBindingSource.EndEdit();
    тестДрайвTableAdapter.Update(бД_автосалонаDataSet);
}

private void checkBox1_CheckedChanged(object sender, EventArgs e)
{
    UpdateTable(sender, e);
}

private void dateTimePicker1_ValueChanged(object sender, EventArgs e)
{
    Console.WriteLine(timedateTimePicker2.Value.TimeOfDay.ToString());
    timeerrorProvider.Clear();
}

private void datedateTimePicker1_ValueChanged(object sender, EventArgs e)
{
    UpdateTable(sender,e);
}

private void button5_Click(object sender, EventArgs e)
{
    this.Validate();
    this.тестДрайвBindingSource.EndEdit();
    try
    {
        тестДрайвTableAdapter.Update(бД_автосалонаDataSet);
    }
    catch (Exception ex)
    {
        MessageBox.Show("Обновление значений невозможно", "ERROOR");
        Console.WriteLine(ex);
        this.тестДрайвTableAdapter.Fill(this.бД_автосалонаDataSet.ТестДрайв);
    }
}

private void vintextBox1_TextChanged(object sender, EventArgs e)
{
    vinerrorProvider.Clear();
}
```

```
        private void fiotextBox1_TextChanged(object sender, EventArgs e)
        {
            fioerrorProvider.Clear();
        }
    }
}
```

5. Форма «Подбор автомобиля»

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace CarsDB
{
    public partial class ConsultantFindCar : Form
    {
        public ConsultantFindCar()
        {
            InitializeComponent();
        }
        public void ConsultantFindCar_FormClosed(object sender, FormClosedEventArgs e)
        {
        }

        private void ConsultantFindCar_Load(object sender, EventArgs e)
        {
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"бД_автосалонаDataSet.CarsNotInDeal". При необходимости она может быть перемещена или
удалена.
            this.carsNotInDealTableAdapter.Fill(this.бД_автосалонаDataSet.CarsNotInDeal);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"бД_автосалонаDataSet.MarksNotInDeal". При необходимости она может быть перемещена или
удалена.
            this.marksNotInDealTableAdapter.Fill(this.бД_автосалонаDataSet.MarksNotInDeal);

        }
        private void FindCar(object sender, EventArgs e)
        {
            string filter="";
            bool marker = false;
            if (marktextBox.Text!="")
            {
                filter = filter + "Марка LIKE'" + marktextBox.Text + "%' ";
                marker = true;
            }
            if (modeltextBox.Text!="")
            {
                if (marker) filter = filter+" AND ";
                filter = filter + "Модель LIKE'" + modeltextBox.Text + "%' ";
                marker = true;
            }
            if (yeartextBox.Text !="")
            {
                if (marker) filter = filter + " AND ";
                filter = filter + "ГодВыпуска ='" + yeartextBox.Text + "' ";
                marker = true;
            }
            if (typecomboBox.Text != "")
```

```
            {
                if (marker) filter = filter + " AND ";
                filter = filter + "ТипКузова ='" + typecomboBox.Text + "' ";
                marker = true;
            }
            if (colorcomboBox.Text != "")
            {
                if (marker) filter = filter + " AND ";
                filter = filter + "Цвет ='" + colorcomboBox.Text + "' ";
                marker = true;
            }
            if(matherialcomboBox.Text!="")
            {
                if (marker) filter = filter + " AND ";
                filter = filter + "МатериалСалона ='" + matherialcomboBox.Text + "' ";
                marker = true;
            }
            if (kppcomboBox.Text != "")
            {
                if (marker) filter = filter + " AND ";
                filter = filter + "КоробкаПередач ='" + kppcomboBox.Text + "' ";
                marker = true;
            }
            if (enginecomboBox.Text != "")
            {
                if (marker) filter = filter + " AND ";
                filter = filter + "ТипДвигателя ='" + enginecomboBox.Text + "' ";
                marker = true;
            }
            if (powerTextBox.Text != "")
            {
                if (marker) filter = filter + " AND ";
                filter = filter + "Мощность ='" + powerTextBox.Text + "' ";
                marker = true;
            }
            if (marker) filter = filter + " AND ";
            filter = filter + "Стоимость <= " + Convert.ToString(pricetrackBar.Value *
1000000)+" ";
            Console.WriteLine(filter);
            carsNotInDealBindingSource.Filter = filter;

        }

        private void label3_Click(object sender, EventArgs e)
        {
        }

        private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
        {



        }

        private void автомобильBindingSource_CurrentChanged(object sender, EventArgs e)
        {

        }

        private void маркаComboBox1_SelectedIndexChanged(object sender, EventArgs e)
        {

            FindCar(sender, e);
        }
```

```
        private void modelsByMarkComboBox_SelectedIndexChanged(object sender, EventArgs e)
        {
            FindCar(sender, e);
        }

        private void автомобильDataGridView_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {

        }

        private void fillBy1ToolStripButton_Click(object sender, EventArgs e)
        {
            try
            {
                this.автомобильTableAdapter.FillBy1(this.бД_автосалонаDataSet.Автомобиль);
            }
            catch (System.Exception ex)
            {
                System.Windows.Forms.MessageBox.Show(ex.Message);
            }

        }




        private void годВыпускаTextBox_TextChanged(object sender, EventArgs e)
        {
            FindCar(sender, e);
        }


        private void comboBox4_SelectedIndexChanged(object sender, EventArgs e)
        {
            FindCar(sender, e);
        }

        private void comboBox5_SelectedIndexChanged(object sender, EventArgs e)
        {
            FindCar(sender, e);
        }

        private void comboBox7_SelectedIndexChanged(object sender, EventArgs e)
        {
            FindCar(sender, e);
        }
        private void comboBox8_SelectedIndexChanged(object sender, EventArgs e)
        {
            FindCar(sender, e);
        }

        private void comboBox1_SelectedIndexChanged_1(object sender, EventArgs e)
        {

        }

        private void pricetrackBar_Scroll(object sender, EventArgs e)
        {
            pricelabel.Text = "Текущая цена " + Convert.ToString(pricetrackBar.Value) + "
млн.";
            FindCar(sender, e);
        }
```

```
        private void marktextBox_TextChanged(object sender, EventArgs e)
        {
            FindCar(sender, e);
        }

        private void modeltextBox_TextChanged(object sender, EventArgs e)
        {
            FindCar(sender, e);
        }

        private void yeartextBox_TextChanged(object sender, EventArgs e)
        {
            FindCar(sender, e);
        }

        private void enginecomboBox_SelectedIndexChanged(object sender, EventArgs e)
        {
            FindCar(sender, e);
        }

        private void matherialcomboBox_SelectedIndexChanged(object sender, EventArgs e)
        {
            FindCar(sender, e);
        }

        private void groupBox1_Enter(object sender, EventArgs e)
        {

        }

        private void pricelabel_Click(object sender, EventArgs e)
        {

        }

        private void pricetrackBar_Scroll_1(object sender, EventArgs e)
        {

        }
    }
}
```

6. Форма «Подбор деталей»

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace CarsDB
{
    public partial class FindDetail : Form
    {
        public FindDetail()
        {
            InitializeComponent();
            detailNotInDealBindingSource.Filter = "Название= '" + nameComboBox.Text + "'";

detailNotInDealTableAdapter.Adapter.Update(бД_автосалонаDataSet.DetailNotInDeal);
```

```
        }

        private void FindDetail_Load(object sender, EventArgs e)
        {
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"бД_автосалонаDataSet.Поставщик". При необходимости она может быть перемещена или удалена.
            this.поставщикTableAdapter.Fill(this.бД_автосалонаDataSet.Поставщик);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"бД_автосалонаDataSet.CountOfDetailsNotInDealByName". При необходимости она может быть
перемещена или удалена.

this.countOfDetailsNotInDealByNameTableAdapter.Fill(this.бД_автосалонаDataSet.CountOfDetail
sNotInDealByName);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"бД_автосалонаDataSet.DetailNotInDeal". При необходимости она может быть перемещена или
удалена.

this.detailNotInDealTableAdapter.Fill(this.бД_автосалонаDataSet.DetailNotInDeal);

        }

        private void clienttextBox_TextChanged(object sender, EventArgs e)
        {

        }

        private void postComboBox_SelectedIndexChanged(object sender, EventArgs e)
        {

        }

        private void gradientPanel2_Paint(object sender, PaintEventArgs e)
        {

        }

        private void nameComboBox_SelectedIndexChanged(object sender, EventArgs e)
        {
            if (posttextBox.Text == "")
            {
                detailNotInDealBindingSource.Filter = "Название= '" + nameComboBox.Text +
"'";

detailNotInDealTableAdapter.Adapter.Update(бД_автосалонаDataSet.DetailNotInDeal);
            }
            else
            {
                detailNotInDealBindingSource.Filter = "Название= '" + nameComboBox.Text +
"' AND Поставщик LIKE'" + posttextBox.Text + "%'";

detailNotInDealTableAdapter.Adapter.Update(бД_автосалонаDataSet.DetailNotInDeal);

            }
        }

        private void posttextBox_TextChanged(object sender, EventArgs e)
        {
            detailNotInDealBindingSource.Filter = "Название= '" + nameComboBox.Text + "'
AND Поставщик LIKE'" + posttextBox.Text + "%'";

detailNotInDealTableAdapter.Adapter.Update(бД_автосалонаDataSet.DetailNotInDeal);
        }
    }
}
```

7. Форма «Создание клиента»

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace CarsDB
{
    public partial class CreateClient : Form
    {
        public CreateClient()
        {
            InitializeComponent();

        }
        public void CreateClient_FormClosed(object sender, FormClosedEventArgs e)
        {
        }

        private void label1_Click(object sender, EventArgs e)
        {

        }

        private void label4_Click(object sender, EventArgs e)
        {

        }

        private void клиентBindingNavigatorSaveItem_Click(object sender, EventArgs e)
        {
            this.Validate();
            this.клиентBindingSource.EndEdit();
            this.tableAdapterManager.UpdateAll(this.бД_автосалонаDataSet);

        }

        private void CreateClient_Load(object sender, EventArgs e)
        {
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"бД_автосалонаDataSet.Клиент". При необходимости она может быть перемещена или удалена.
            this.клиентTableAdapter.Fill(this.бД_автосалонаDataSet.Клиент);

        }

        private void фИОTextBox_TextChanged(object sender, EventArgs e)
        {
            errorProvider1.Clear();
        }

        private void номерТелефонаTextBox_TextChanged(object sender, EventArgs e)
        {
            errorProvider2.Clear();
        }
        private void паспортныеДанныеTextBox_TextChanged(object sender, EventArgs e)
        {
            errorProvider3.Clear();
        }
```

```
        private void фИОLabel_Click(object sender, EventArgs e)
        {

        }

        private void номерТелефонаLabel_Click(object sender, EventArgs e)
        {

        }

        private void label1_Click_1(object sender, EventArgs e)
        {

        }

        private void button1_Click(object sender, EventArgs e)
        {
            if (textBox1.Text!="" && textBox2.Text == "")
                клиентBindingSource.Filter = "ФИО LIKE'" + textBox1.Text + "%'";
            if (textBox1.Text == "" && textBox2.Text != "")
                клиентBindingSource.Filter = "НомерТелефона LIKE'" + textBox2.Text + "%'";
            if (textBox1.Text != "" && textBox2.Text != "")
                клиентBindingSource.Filter = "НомерТелефона LIKE'" + textBox2.Text + "%'
AND ФИО LIKE'" + textBox1.Text + "%'";

        }

        private void textBox1_TextChanged(object sender, EventArgs e)
        {
            if((клиентBindingSource.Filter=="")||(textBox2.Text==""))
                клиентBindingSource.Filter = "ФИО LIKE'" + textBox1.Text + "%'";
            else
                клиентBindingSource.Filter = "НомерТелефона LIKE'" + textBox2.Text + "%'
AND ФИО LIKE'" + textBox1.Text + "%'";
        }

        private void textBox2_TextChanged(object sender, EventArgs e)
        {
            if ((клиентBindingSource.Filter == "") || (textBox1.Text == ""))
                клиентBindingSource.Filter = "НомерТелефона LIKE'" + textBox2.Text + "%'";
            else
                клиентBindingSource.Filter = "НомерТелефона LIKE'" + textBox2.Text + "%'
AND ФИО LIKE'" + textBox1.Text + "%'";
        }

        private void button2_Click(object sender, EventArgs e)
        {
            if (фИОTextBox.Text == "") { errorProvider1.SetError(фИОTextBox, "Введите
ФИО"); return; }
            if (номерТелефонаTextBox.Text == "") {
errorProvider2.SetError(номерТелефонаTextBox, "Введите номер телефона"); return; }
            if (паспортныеДанныеTextBox.Text == ""){
errorProvider3.SetError(паспортныеДанныеTextBox, "Введите паспортные данные"); return;}
            int? error = 0;
            клиентTableAdapter.InsertClient(фИОTextBox.Text, номерТелефонаTextBox.Text,
паспортныеДанныеTextBox.Text, ref error);
            if (error == 0)
            {
                MessageBox.Show("Выполнено успешно", "Добавление клиента");
                номерТелефонаTextBox.Text = "";
                фИОTextBox.Text = "";
                паспортныеДанныеTextBox.Text = "";
                this.клиентTableAdapter.Fill(this.бД_автосалонаDataSet.Клиент);
            }
```

```
            if(error==1)
            {
                errorProvider1.SetError(фИОTextBox, "Существует клиент с таким ФИО");
            }
            if (error == 2)
            {
                errorProvider2.SetError(номерТелефонаTextBox, "Существует клиент с таким
номером телефона");
            }
            if (error == 3)
            {
                errorProvider3.SetError(паспортныеДанныеTextBox, "Существует клиент с
такими паспортными данными");
            }
        }

        private void gradientPanel1_Paint(object sender, PaintEventArgs e)
        {

        }
    }
}
```

8. Форма «Домашняя страница: Продавец»

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace CarsDB
{
    public partial class SalesmanHome : Form
    {
        string position;
        public SalesmanHome(string pos)
        {
            InitializeComponent();
            this.position = pos;
            this.Text = this.Text +" "+ this.position;
            sotrtextBox.Text = this.position;
        }
        public void SalesmanHome_FormClosed(object sender, FormClosedEventArgs e)
        {
            Form frm = Application.OpenForms[0];
            frm.Show();
        }
        private void SalesmanHome_Load(object sender, EventArgs e)
        {
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"бД_автосалонаDataSet.ДопОборудованиеСделки". При необходимости она может быть перемещена
или удалена.

this.допОборудованиеСделкиTableAdapter.Fill(this.бД_автосалонаDataSet.ДопОборудованиеСделки
);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"бД_автосалонаDataSet.ДопОборудование". При необходимости она может быть перемещена или
удалена.

this.допОборудованиеTableAdapter.Fill(this.бД_автосалонаDataSet.ДопОборудование);
```

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

this.detailNotInDealTableAdapter.Fill(this.бД_автосалонаDataSet.DetailNotInDeal);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"бД_автосалонаDataSet.Клиент". При необходимости она может быть перемещена или удалена.
            this.клиентTableAdapter.Fill(this.бД_автосалонаDataSet.Клиент);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"бД_автосалонаDataSet.CarsNotInDeal". При необходимости она может быть перемещена или
удалена.
            this.carsNotInDealTableAdapter.Fill(this.бД_автосалонаDataSet.CarsNotInDeal);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"бД_автосалонаDataSet.Сделка". При необходимости она может быть перемещена или удалена.
            this.сделкаTableAdapter.Fill(this.бД_автосалонаDataSet.Сделка);
           //
this.detailsOfDealTableAdapter.Fill(this.бД_автосалонаDataSet.DetailsOfDeal,Convert.ToInt32
(dealnumtextBox.Text));

        }

        private void tabPage1_Click(object sender, EventArgs e)
        {

        }

        private void tabPage1_Click_1(object sender, EventArgs e)
        {

        }

        private void сделкаBindingNavigatorSaveItem_Click(object sender, EventArgs e)
        {
            this.Validate();
            this.сделкаBindingSource.EndEdit();
            this.tableAdapterManager.UpdateAll(this.бД_автосалонаDataSet);

        }

        private void button1_Click(object sender, EventArgs e)
        {
            if (vintextBox.Text == "")
            {
                vinerrorProvider.SetError(vintextBox, "Введите VIN");
                return;
            }
            if (vintextBox.TextLength < 17)
            {
                vinerrorProvider.SetError(vintextBox, "Минимальная длина VIN 17");
                return;
            }
            if (pricetextBox.Text == "")
            {
                priceerrorProvider.SetError(pricetextBox, "Введите сумму сделки");
                return;
            }
            int? error = 0;
            DateTime date = new DateTime();
            Console.WriteLine(date.Date);
            сделкаTableAdapter.InsertDeal(this.position, clienttextBox.Text,
vintextBox.Text, DateTime.Now.Date, Convert.ToInt32(pricetextBox.Text), ref error);
            if (error == 0)
            {
                MessageBox.Show("Выполнено успешно!");
                сделкаTableAdapter.Fill(this.бД_автосалонаDataSet.Сделка);
```

```
            }
            if (error == 1)
            {
                vinerrorProvider.SetError(vintextBox, "Данный автомобиль уже в сделке или
его не существует");
                return;
            }
            if (error == 3)
            {
                clientProvider.SetError(clienttextBox, "Не существует такого клиента");
            }
            if (error == 2)
            {
                priceerrorProvider.SetError(pricetextBox, "Сумма сделки меньше стоимости
автомобиля");
            }
        }

        private void vintextBox_TextChanged(object sender, EventArgs e)
        {
            vinerrorProvider.Clear();
            decimal? price = 0;
            сделкаTableAdapter.PriceByVIN(vintextBox.Text, ref price);
            price=Convert.ToInt32(price);
            pricetextBox.Text = price.ToString();
        }

        private void pricetextBox_TextChanged(object sender, EventArgs e)
        {
            priceerrorProvider.Clear();
        }

        private void detailNotInDealDataGridView_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {
        }
        private void detailNotInDealDataGridView_CellClick(object sender,
DataGridViewCellEventArgs e)
        {
            detailnumtextBox1.Text =
detailNotInDealDataGridView.CurrentRow.Cells[0].Value.ToString();
        }



        private void searchtextBox_TextChanged(object sender, EventArgs e)
        {
            detailNotInDealBindingSource.Filter = "Название LIKE'" + searchtextBox.Text +
"%'";
            detailNotInDealTableAdapter.Adapter.Update(бД_автосалонаDataSet);
        }

        private void searchfiotextBox_TextChanged(object sender, EventArgs e)
        {
            сделкаBindingSource.Filter = "ФиоКлиента LIKE'" + searchfiotextBox.Text + "%'";
            сделкаTableAdapter.Update(бД_автосалонаDataSet);
        }

        private void button2_Click(object sender, EventArgs e)
        {
            сделкаBindingSource.RemoveCurrent();
            this.Validate();
            this.сделкаBindingSource.EndEdit();
            сделкаTableAdapter.Update(бД_автосалонаDataSet);
```

```
this.detailNotInDealTableAdapter.Fill(this.6Д_автосалонаDataSet.DetailNotInDeal);
        }

        private void button3_Click(object sender, EventArgs e)
        {
            this.Validate();
            this.сделкаBindingSource.EndEdit();
            try
            {
                сделкаTableAdapter.Update(6Д_автосалонаDataSet);
            }
            catch (Exception ex)
            {
                MessageBox.Show("Обновление значений невозможно", "Обновление сделки");
                Console.WriteLine(ex);
                this.сделкаTableAdapter.Fill(this.6Д_автосалонаDataSet.Сделка);
            }
        }

        private void clienttextBox_TextChanged(object sender, EventArgs e)
        {
            clientProvider.Clear();
        }

        private void button4_Click(object sender, EventArgs e)
        {
            Console.WriteLine( сделкаDataGridView.CurrentRow.Cells[0].Value.ToString());
            if (dealnumtextBox.Text=="")
            {
                MessageBox.Show("Не выбрана сделка", "Ошибка добавления детали");
                return;
            }
            if(detailnumtextBox1.Text=="")
            {
                MessageBox.Show("Не выбрана деталь", "Ошибка добавления детали");
                return;
            }
            допОборудованиеСделкиTableAdapter.Insert(Convert.ToInt32(dealnumtextBox.Text),
Convert.ToInt32(detailnumtextBox1.Text));

this.detailNotInDealTableAdapter.Fill(this.6Д_автосалонаDataSet.DetailNotInDeal);
            this.detailsOfDealTableAdapter.Fill(this.6Д_автосалонаDataSet.DetailsOfDeal,
Convert.ToInt32(dealnumtextBox.Text));


        }

        private void сделкаDataGridView_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {
        }
        private void сделкаDataGridView_CellClick(object sender, DataGridViewCellEventArgs
e)
        {
            dealnumtextBox.Text = сделкаDataGridView.CurrentRow.Cells[0].Value.ToString();
            this.detailsOfDealTableAdapter.Fill(this.6Д_автосалонаDataSet.DetailsOfDeal,
Convert.ToInt32(dealnumtextBox.Text));
        }

        private void dealnumtextBox_TextChanged(object sender, EventArgs e)
        {
            if(dealnumtextBox.Text!="")
            {
```

```
this.detailsOfDealTableAdapter.Fill(this.6Д_автосалонаDataSet.DetailsOfDeal,
Convert.ToInt32(dealnumtextBox.Text));
            }
        }

        private void button5_Click(object sender, EventArgs e)
        {
        }

        private void button6_Click(object sender, EventArgs e)
        {
            Form frm = new ConsultantFindCar();
            frm.Show();
        }

        private void button7_Click(object sender, EventArgs e)
        {
            Form frm = new CreateClient();
            frm.Show();
        }

        private void button8_Click(object sender, EventArgs e)
        {
            int? detail_num =
Convert.ToInt32(detailsOfDealDataGridView.CurrentRow.Cells[0].Value.ToString());
            this.допОборудованиеСделкиTableAdapter.DeleteDetailFromDeal(detail_num);

this.detailNotInDealTableAdapter.Fill(this.6Д_автосалонаDataSet.DetailNotInDeal);
            this.detailsOfDealTableAdapter.Fill(this.6Д_автосалонаDataSet.DetailsOfDeal,
Convert.ToInt32(dealnumtextBox.Text));
        }

        private void gradientPanel2_Paint(object sender, PaintEventArgs e)
        {

        }

        private void gradientPanel2_Paint_1(object sender, PaintEventArgs e)
        {

        }
    }
}
```

9. Форма «Домашняя страница: Механик»

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace CarsDB
{
    public partial class MechanicEquipment : Form
    {
        public MechanicEquipment()
        {
```

```
            InitializeComponent();

        }
        public void MechanicEquipment_FormClosed(object sender, FormClosedEventArgs e)
        {
            Form frm = Application.OpenForms[0];
            frm.Show();
        }
        private void fillToolStripButton_Click(object sender, EventArgs e)
        {

        }

        private void radioButton1_CheckedChanged(object sender, EventArgs e)
        {
            searchtextBox.MaxLength = 32767;
        }

        private void autoradioButton_CheckedChanged(object sender, EventArgs e)
        {
            searchtextBox.MaxLength = 17;
        }

        private void button1_Click(object sender, EventArgs e)
        {
            if (searchtextBox.Text == "")
            {
                errorProvider1.SetError(searchtextBox, "Введите данные");
                return;
            }
            if (dealradioButton.Checked)
            {
                dataGridView1.DataSource = carEquipmentBindingSource;
                carEquipmentTableAdapter.Fill(this.бД_автосалонаDataSet.CarEquipment,
Convert.ToInt32(searchtextBox.Text));
            }
            else
            {
                if (searchtextBox.Text.Length < 17) errorProvider1.SetError(searchtextBox,
"Введите корректный VIN");
                dataGridView1.DataSource = carEquipmentByCarBindingSource;

carEquipmentByCarTableAdapter.Fill(this.бД_автосалонаDataSet.CarEquipmentByCar,
searchtextBox.Text);
            }

        }

        private void MechanicEquipment_Load(object sender, EventArgs e)
        {
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"бД_автосалонаDataSet.ДопОборудование". При необходимости она может быть перемещена или
удалена.

this.допОборудованиеTableAdapter.Fill(this.бД_автосалонаDataSet.ДопОборудование);

        }

        private void searchtextBox_TextChanged(object sender, EventArgs e)
        {
            errorProvider1.Clear();
        }
    }
}
```