Оглавление

[Введение 3](#_Toc120213077)

[Глава 1. Проектирование структуры базы данных 5](#_Toc120213078)

[1.1. Анализ предметной области 5](#_Toc120213079)

[1.2. Концептуальное проектирование 14](#_Toc120213080)

[1.3. Проектирования БД автосервиса 18](#_Toc120213081)

[1.4. Физическое проектирование 24](#_Toc120213082)

[Глава 2. Описание функционирования БД 34](#_Toc120213083)

[2.1. Назначение и перечень функций базы данных 34](#_Toc120213084)

[2.2. Описание работы с базой данных 36](#_Toc120213085)

[Заключение 69](#_Toc120213086)

[Список используемых источников 71](#_Toc120213087)

[Приложение А. Примеры заполнения таблиц 72](#_Toc120213088)

[Приложение Б. SQL-операторы создания объектов БД 78](#_Toc120213089)

[Приложение B. Код программы 92](#_Toc120213090)

Введение

Актуальность выбранной темы заключается в необходимости осмысления современной структуры данных, поскольку сейчас, все чаще люди перекладывают обязанности на компьютеры, разрабатывая программы, непосредственно связанные с обучением.

Современные тренды программирования тесно связаны не только с визуальной составляющей, но и с пользовательским опытом пользователей ресурса, их удовлетворенности понятным и лаконичным дизайном, интуитивными настройками. Все это делает работу программ важной, ведь их создание требует учета сочетание визуальной идентификации и программного обеспечения, используя и проектируя стилевую модель для каждого проекта отдельно.

Целью курсовой работы является теоретическая разработка структуры хранения данных автосервиса, а также реализация готового программного средства, обеспечивающего полноценный функционал для всех участников бизнес-процессов.

Для реализации цели исследования были поставлены следующие задачи:

* Проанализировать предметную область
* Провести концептуальное проектирование
* Спроектировать БД автосервиса
* Провести физическое проектировании
* Спроектировать программное средство автосервиса

Объектом исследования является процесс разработки базы данных и программного продукта.

Предметом исследования является процесс проектирования базы данных и элементов программного средства.

Практическое значение полученных результатов заключается в том, что в рамках данной курсовой работы разработана программа, которая автоматизирует работу автосервиса. Кроме того, будет проанализирована и систематизирована теоретическая информация относительно имеющихся подходов и алгоритмов проектирования БД и построения программного средства, реализующего заложенный функционал.

Глава 1. Проектирование структуры базы данных

1.1. Анализ предметной области

Предметная область — это целенаправленная первичная трансформация картины внешнего мира в некоторую умозрительную картину, определенная часть которой фиксируется в ИС в качестве алгоритмической модели фрагмента действительности[1].

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. В описании предметной области должны присутствовать характеристики возможных воздействий окружающей среды на элементы и явления предметной области, а также обратные воздействия этих элементов и явлений на среду[2].

Специфика предметной области может оказывать существенное влияние на характер функционирования проектируемой системы.

Предметную область можно определить как объект или производственную систему со всем комплексом понятий и знаний о ее функционировании. При исследовании проблемной области необходимы знания о задачах, решаемых в проектируемой системе, и стоящих перед ней целях[2].

Одна из первых задач, с решением которых сталкивается разработчик базы данных — это изучение, осмысление и анализ предметной области. Дело в том, что предметная область сильно влияет на все аспекты проектирования базы данных: требования к системе, количество конечных пользователей системы, модель хранения данных, реализацию и т.д.

Анализ предметной области позволяет выделить ее сущности, определить первоначальные требования к функциональности и определить границы проекта. Модель предметной области должна быть документирована, храниться и поддерживаться в актуальном состоянии до этапа реализации.

Предметной областью в данной самостоятельной работе является деятельность автосервиса.

Автосервис предоставляет своему клиенту широкий спектр услуг по ремонту, восстановлению и дооснащению индивидуальных средств передвижения. Также практически каждый автосервис предоставляет услуги заказа, подбора и покупки необходимых расходных материалов и деталей для проведения соответствующих работ. Обслуживание клиента как правило происходит в несколько этапов, на каждом из которых заказчик получает соответствующий “заказ-наряд”.

Ручное оформление схожего рода документов является нецелесообразным из-за больших временных затрат.

В связи с чем актуальность применении баз данных в автосервисе обусловлена повышением эффективности предоставления услуг. В настоящее время автосервисы страны массово проводят цифровизацию своего бизнеса с внедрением баз данных.

Далее рассмотрим имеющиеся специализированные программные продукты российских разработчиков, созданные для автоматизации процессов функционирования автосервиса.

Бесспорным лидером отечественной отрасли по разработке программных решений для бизнеса является компания 1С, разработавшая программное средство “1С:Предприятие 8. Автосервис”[3].

Учитывая реальную специфику работы автосервиса, программное средство “1С:Предприятие 8. Автосервис” представляет следующий перечень возможностей:

* Функции CRM-системы: ведение данных клиентов, предварительная запись на ремонт, фиксация всех контактов с клиентами, анализ клиентской базы.
* Учет выполнения ремонта автомобиля с формированием необходимых документов: акт осмотра, рабочий лист, заказ-наряд, квитанция к заказ-наряду. (рис. 1.1)



Рис.1.1 Заказ-наряд “1С:Предприятие 8. Автосервис”

* Поэтапное выполнение ремонта (заявка, выполнение ремонта, ожидание, ремонт выполнен, заказ-наряд закрыт).
* Проведение платных и бесплатных видов ремонта, гибкая скидочная система. (рис. 1.2)
* Гибкое ценообразование по работам в зависимости от модели или комплектации автомобиля, контрагента, вида и места ремонта.

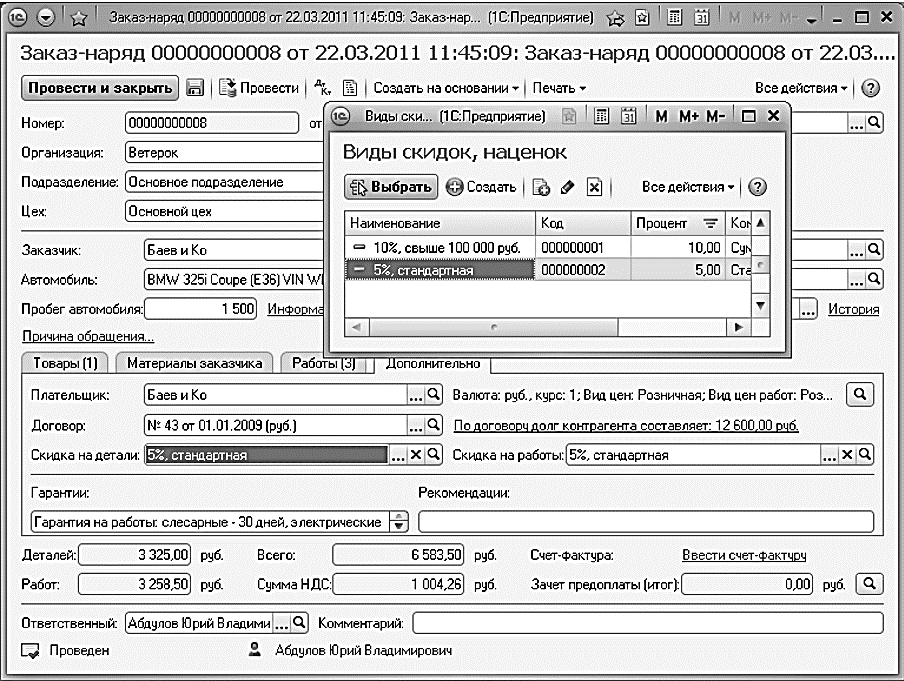


Рис. 1.2. Виды скидок “1С:Предприятие 8. Автосервис”

Дополнительно заявлен следующий перечень функции, реализованных в программном средстве от компании 1С:

* Справочник “Номенклатура”. Для запчастей организован учет аналогов, доступных по ссылке непосредственно из карточки запчасти (рис. 1.3)

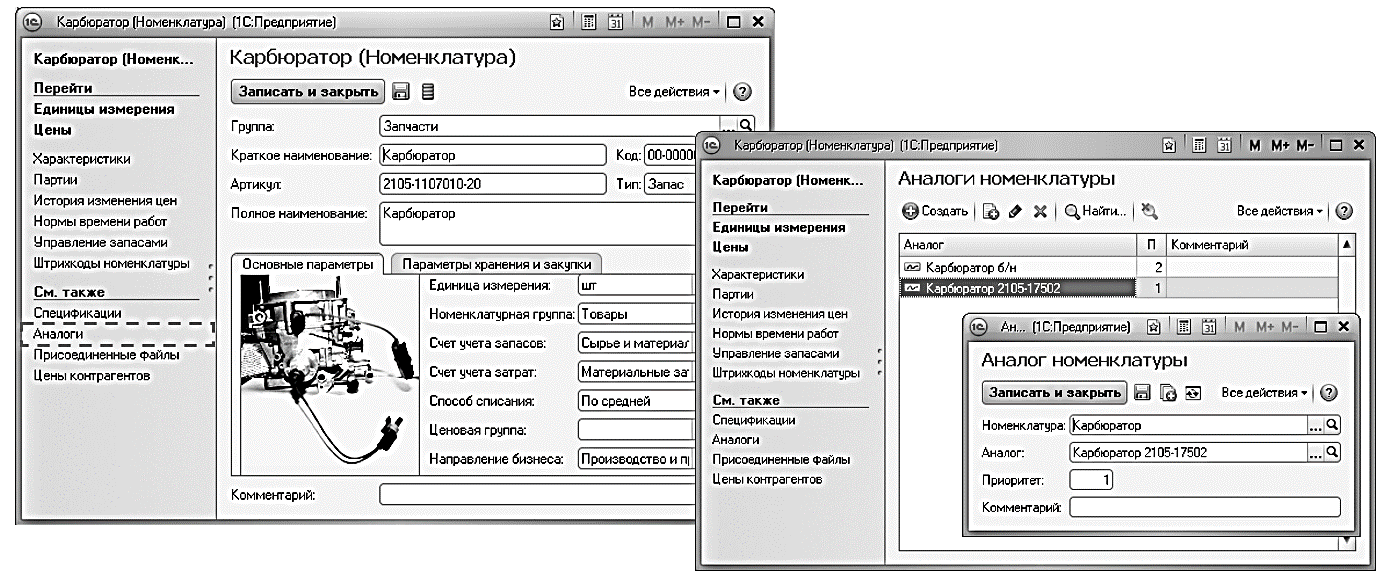


Рис. 1.3. Номенклатура “1С:Предприятие 8. Автосервис”

* Полный цикл операций по учету запчастей: заказ покупателя, заказ поставщику, поступление, возврат, резервирование, реализация (опт и розница), инвентаризация и списание.
* Отражение всех перемещений запасных частей в процессе ремонта: выдача со склада под заказ-наряд, возврат на склад, перемещения между заказ-нарядами и др.
* Учет запчастей в цехах автосервиса по незакрытым заказ-нарядам (незавершенное производство).
* Анализ деятельности: формирование различных аналитических отчетов, расчет фактической себестоимости и рентабельности, оценка эффективности работы персонала, финансовый результат (рис. 1.4).

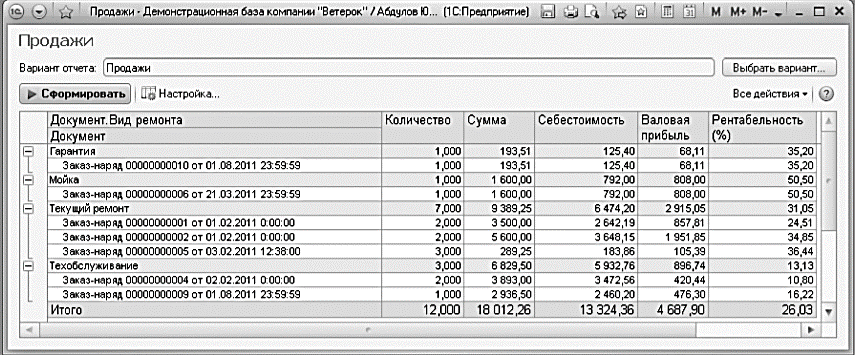


Рис. 1.4. Отчет о продажах “1С:Предприятие 8. Автосервис”

Вторым программным продуктом в рассматриваемой предметной области является программное решение от компании ПростойСофт, названное “ПростойСофт Автосервис”[4].

В данной программе разработчиками заявлен следующий перечень функциональных возможностей:

* Ведение клиентов и их контактных данных (рис. 1.5).
* Учет оказанных услуг и ремонтных работ.
* Ведение предварительной записи на ремонтные работы.

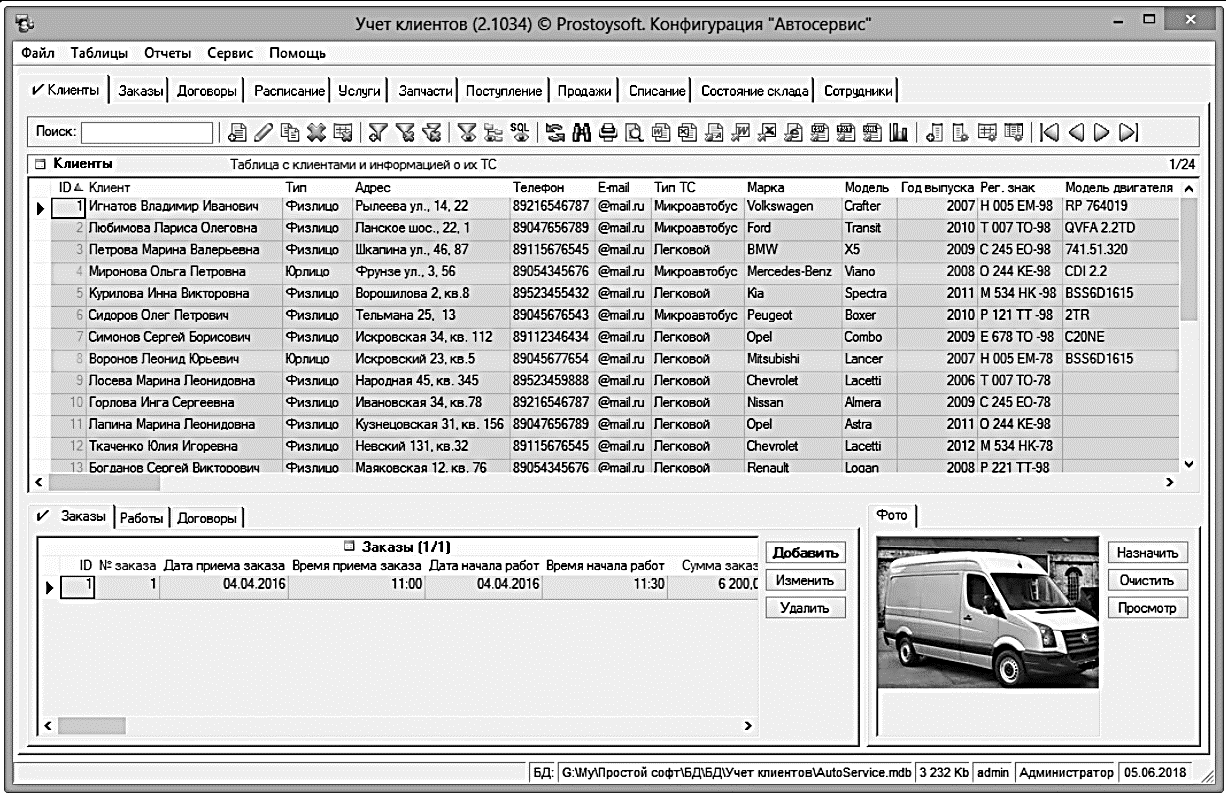


Рис. 1.5. Клиенты “ПростойСофт Автосервис”

* Ведение справочника запчастей.
* Формирование договоров, заказ-нарядов (рис. 1.6), актов и прочих документов.

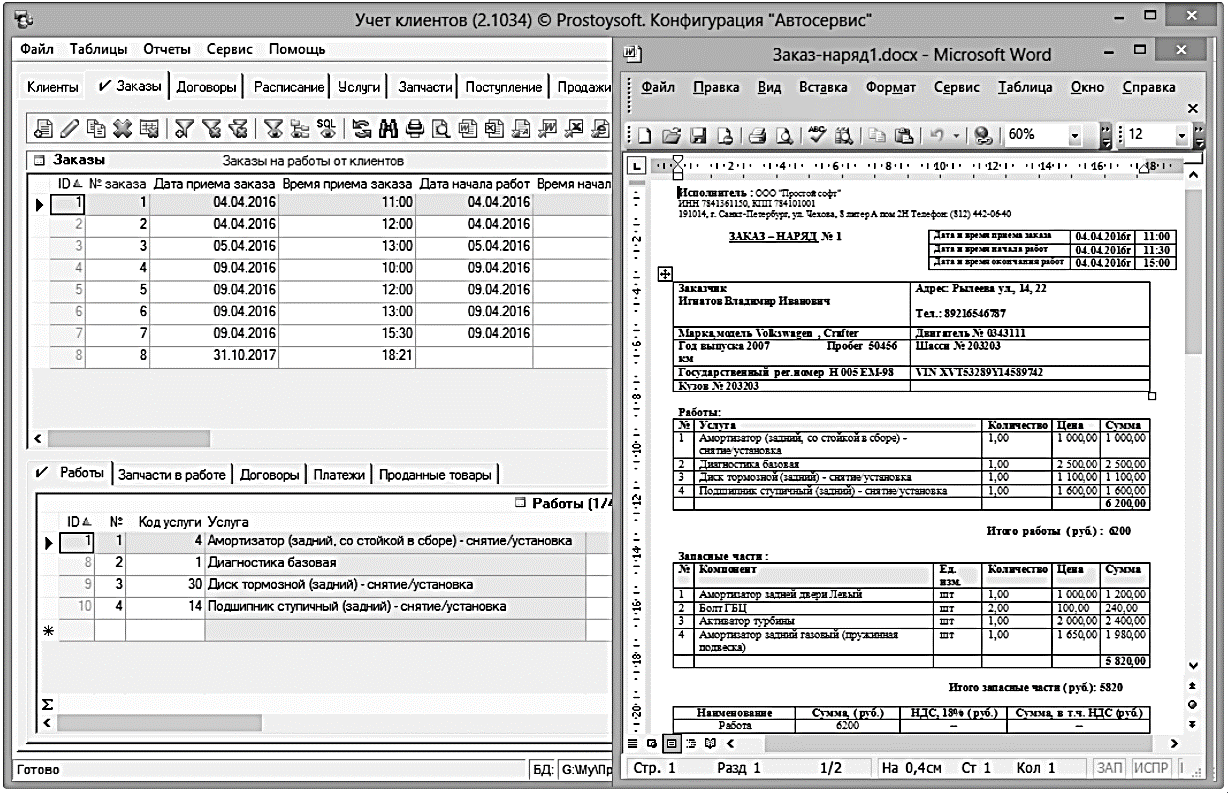


Рис. 1.6. Заказ-наряды “ПростойСофт Автосервис”

* Набор стандартных отчетов с возможностью создания новых.
* Регистрация и учет заявок на ремонт автомобиля.
* Учет затраченных запчастей на ремонт (рис. 1.7).
* Ведение справочника услуг автосервиса.
* Импорт и экспорт данных.

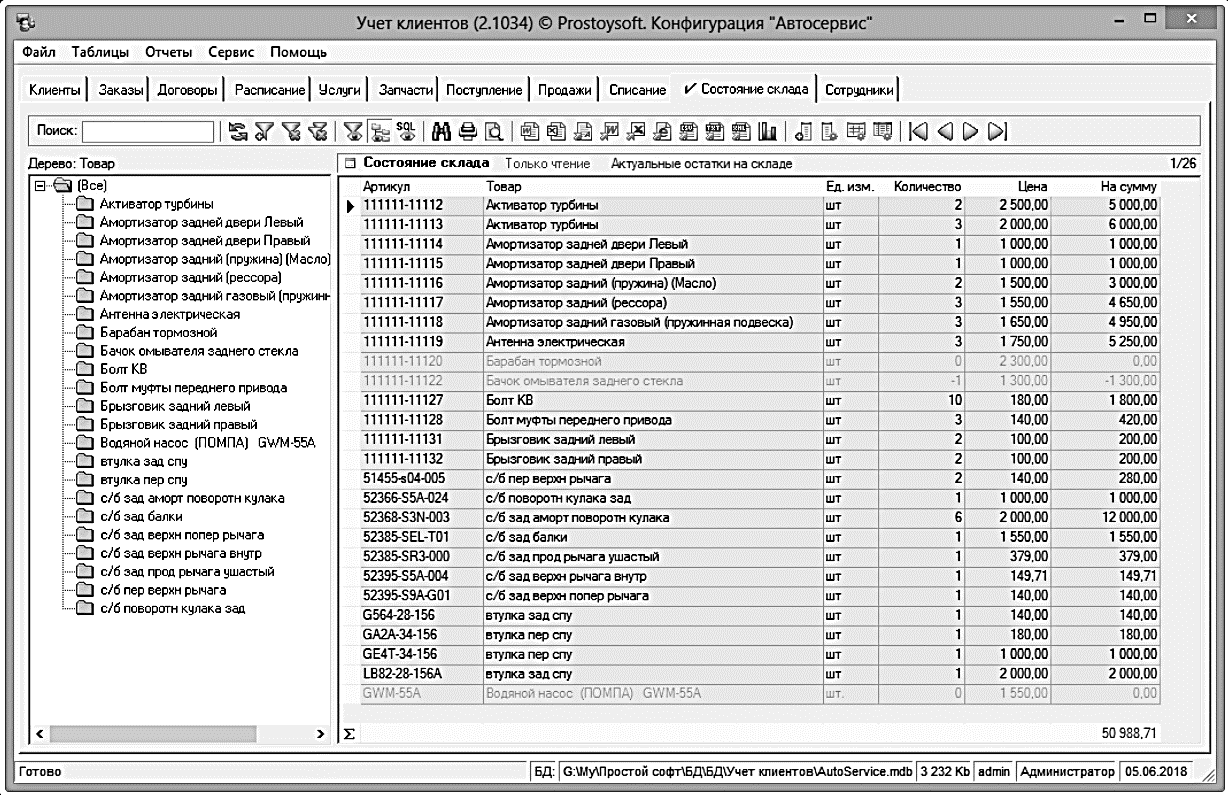


Рис. 1.7. Состояние склада “ПростойСофт Автосервис”

“ПростойСофт Автосервис” удобна для автоматизации работы автосервиса. Данная программа позволяет руководству усовершенствовать процесс обслуживания клиентов благодаря ведению всестороннего учета. Благодаря возможности ведения базы клиентов их контактной информации, ведения справочников запчастей и услуг автосервиса, формирование предварительной записи на ремонт руководству автосервиса удастся уменьшить издержки.

Обе рассмотренные в рамках данной главы программы предоставляют пользователю схожий широкий функционал.

Базовый функционал программ по созданию заказ-нарядов, хранению информации о клиента и их персональных средствах перемещения имеют схожий формат и конечную реализацию. Функционал продаж также реализован в двух программных средствах.

Сравнивая два программных продукта, можно заметить, что “ПростойСофт Автосервис”, в отличии от “1С:Предприятие 8. Автосервис”, разбивает все функции на отдельные модули, которые не настолько тесно связаны с другими функциями. “1С:Предприятие 8. Автосервис”, напротив, в своем программном продукте применяет более комплексный подход, объединяя функционал в единую информационную систему, где модули взаимосвязаны друг с другом.

Также в “1С:Предприятие 8. Автосервис” имеются удобные с точки зрения пользовательского опыта дополнительные функции, такие как “Создать на основании”, “Подбор”, “Сортировать”.

Оба программных средства содержат базы данных прикладной специфики, что позволяет создавать новые таблицы, отчеты, графики, добавлять поля, формировать списки и многое другое. Программы учета интуитивно просты и понятны, их использование не требует специальных знаний и квалификации в ИТ-сфере.

Оба программных средства легко и быстро настраиваются под индивидуальные требования.

В рамках самостоятельной реализации программного средства для автосервиса планируется реализовать следующие функции:

* Создание заказ-нарядов.
* Учет затраченных запчастей на ремонт.
* Учет основных торговых операций – приход, продажа и списание запчастей.
* Ведение справочника услуг автосервиса.
* Ведение клиентов и их контактных данных.
* Учет оказанных услуг и ремонтных работ.

Конечный список функциональных возможностей может изменяться и дополняться в процессе непосредственной реализации программного средства.

1.2. Концептуальное проектирование

Концептуальное проектирование заключается в формализованном описании предметной области. Целью этапа концептуального проектирования является определение содержания базы данных[5].

Диаграмма вариантов использования (ДВИ) является исходным концептуальным представлением системы в процессе ее проектирования и разработки. Данная диаграмма состоит из потенциальных пользователей системы (действующих лиц), вариантов использования и отношений между ними. При построении диаграммы могут использоваться также общие элементы нотации: примечания и механизмы расширения.

Суть данной диаграммы состоит в следующем: проектируемая система представляется в виде множества действующих лиц, взаимодействующих с системой с помощью так называемых вариантов использования. При этом действующим лицом называется любой объект, субъект или система, взаимодействующая с моделируемой системой извне. В свою очередь вариант использования – это спецификация функций, которые система предоставляет действующему лицу. Другими словами, каждый вариант использования определяет некоторый набор действий, совершаемых системой при взаимодействии с потенциальным пользователем. При этом в модели никак не отражается то, каким образом будет реализован этот набор действий.

База данных предназначена для осуществления деятельности автосервиса.

По телефону и/или при личном визите владелец индивидуального транспортного средства может получить информацию о стоимости ремонта, о перечне оказываемых услуг и о времени, когда автосервис сможет принять транспортное средство для диагностики и проведения соответствующих работ.

В случае, когда клиент предварительно позвонил в автосервис менеджер возьмет необходимую информацию для внесения клиента в базу данных и запишет его на диагностику.

В случае непосредственного приезда в автосервис владелец будет контактировать со свободным механиком, который проведет диагностику транспортного средства и в последствии создаст заказ-наряд на ремонт.

В реально функционирующих автосервисах работает коллектив, состоящий из большого количества профильных специалистов. В целях упрощения будущей модели программного средства весь штат разнопрофильных мастеров автосервиса примем за универсального механика.

Также в реально функционирующих автосервисах существует отдел менеджеров, занимающихся закупками запчастей и их реализацией в ходе ремонта.

В результате анализа было решено, что с проектируемой системой будут взаимодействовать:

* Директор.
* Менеджер.
* Механик.

В реальной жизни клиент автосервиса может просматривать электронный каталог услуг и статус выполнения работ с его транспортным средством. Однако в проектируемой системе такой возможности у него нет, поэтому клиент автосервиса не будет являться действующим лицом.

В случае, если клиенту автосервиса необходимо узнать о состоянии ремонта его транспортного средства он может связаться с менеджером и уточнить соответствующую информацию.

Рассмотрим варианты использования каждого из действующих лиц.

Механику доступны следующие функции:

* Диагностика ТС.
* Создание заказ-нарядов.
* Обновление состояния выполнения ремонтных работ.
* Фиксация затраченных запчастей в ходе ремонта.
* Заказ необходимых запчастей.
* Подбор запчастей аналогов.

Менеджеру доступны следующие функции:

* Ведение клиентов и их контактных данных.
* Учет оказанных услуг и ремонтных работ.
* Запись клиентов/аннулирование записи.
* Выдача информации об оказываемых услугах.

Директору доступны следующие функции:

* Ведение справочника услуг автосервиса.
* Учет сотрудников
* Учет заказанных\хранимых деталей на складе
* Составим общую диаграмму вариант использования (рис 1.8).

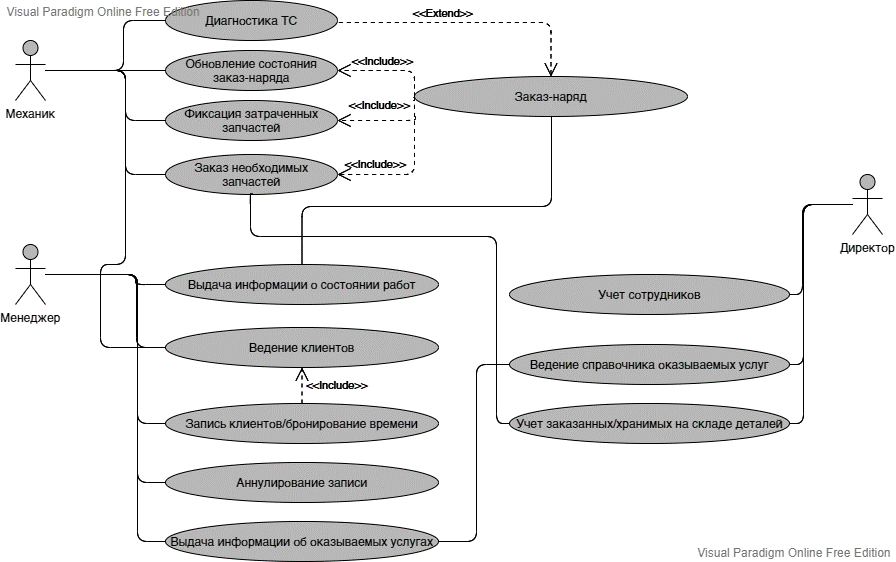


Рис. 1.8. Диаграмма вариантов использования БД автосервис

Конечный список функциональных возможностей может изменяться и дополняться в процессе непосредственной реализации программного средства.

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

* Данные клиентов автосервиса, которые включают в себя его персональные данные
* Данные о транспортных средствах клиентов
* Наличие запчастей на складе.
* Регистрация заказа и поступления запчастей на склад
* Регистрация списания запчастей в ходе ремонта.
* Временную сетку с рабочими часами мастеров автосервиса.
* Сведения о записи на определенное временя клиентом с возможностью отмены соответствующей записи
* Сведения о перечне предоставляемых автосервисом услуг
* Сведения о работниках, данные о всех вакансиях (должность, график работы, заработная плата, образование, опыт работы и т.п.).

Необходимые данные можно классифицировать по частоте их изменения:

− условно-постоянные данные (например, сведения о работниках);

− данные, которые оперативно обновляются при каждом решении задачи (когда и какие запчасти поступили на склад, когда их списали, на какое время записался клиент).

1.3. Проектирования БД автосервиса

**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. Его цель состоит в создании логической модели данных. Концептуальная модель данных, созданная на предыдущем этапе, уточняется и преобразуется в логическую модель данных[6].

Схема «сущность-связь» (также ERD или ER-диаграмма) — это разновидность блок-схемы, где показано, как разные «сущности» (люди, объекты, концепции и так далее) связаны между собой внутри системы. ER-диаграммы чаще всего применяются для проектирования и отладки реляционных баз данных в сфере образования, исследования и разработки программного обеспечения и информационных систем для бизнеса. ER-диаграммы (или ER-модели) полагаются на стандартный набор символов, включая прямоугольники, ромбы, овалы и соединительные линии, для отображения сущностей, их атрибутов и связей. Эти диаграммы устроены по тому же принципу, что и грамматические структуры: сущности выполняют роль существительных, а связи — глаголов.

Цель логического проектирования 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 или сетевая модель).

Предметная область – автомастерская, содержит информацию о работниках (механик, менеджер, директор), заказ-нарядах, транспортных средствах, клиентах, деталях, заказах, услугах, рабочих днях, записях клиентов.

Построим ER-диаграммы всех сущностей и связей между ними.

Каждый механик может диагностировать несколько транспортных средств, а может и не заниматься диагностикой. Каждое транспортное средство должно быть продиагностировано одним механиком (рис 1.9).



Рис. 1.9. ER-диаграмма отношения Диагностирует

Каждый механик может создать несколько заказ-нарядов, а может и не заниматься созданием заказ-нарядов. Каждый заказ-наряд должен быть создан одним механиком (рис. 1.10).



Рис. 1.10. ER-диаграмма отношения Создает

Каждый работник, выполняя свои должностные обязанности, выходит на смены, работая в определенные дни. Таким образом, каждый работник работает во многие дни. Каждый день на работу выходит несколько сотрудников. (рис 1.11).



Рис. 1.11. ER-диаграмма отношения Работает

В каждом заказ-наряде могут расходоваться несколько деталей. Детали могут не расходоваться в заказ-нарядах (рис. 1.12).



Рис. 1.12. ER-диаграмма отношения Затрачивается

Каждая деталь может иметь множество деталей аналогов (рис 1.13) или ни одного.

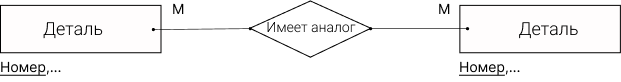


Рис. 1.13. ER-диаграмма отношения Имеет аналог

Каждое транспортное средство относится к нескольких заказ-нарядам, а заказ-наряд несущественен без транспортного средства (рис. 1.14).



Рис. 1.14. ER-диаграмма отношения Фигурирует

Каждый клиент может несколько раз записаться в автосервис или не записываться ни разу, а запись обязательно должна быть оформлена на конкретного клиента (рис. 1.15).



Рис. 1.15. ER-диаграмма отношения Запрашивает

Каждый клиент владеет по меньшей мере одним транспортным средством, в противном случае ему нет необходимости обращаться в автосервис. А транспортное средство имеет одного владельца (рис. 1.16).



Рис. 1.16. ER-диаграмма отношения Владеет

Несколько услуг могут фигурировать (рис. 1.17) в рамках одного заказ-наряда или не фигурировать. Заказ-наряд не может быть пустым и обязан содержать хотя бы одну услугу.



Рис. 1.17. ER-диаграмма отношения Предоставяется

Сформулируем набор предварительных отношений с указанием предполагаемоего первичного ключа для каждого отношения.

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

1. Работник (ФИО, …)
2. Транспортное средство (WIN-номер, ФИОдиагност, …) – добавился неключевой атрибут ФИОдиагност.

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

1. Работник (ФИО, …)
2. Заказ-наряд (Номер, ФИОсозд, …) – добавился неключевой атрибут ФИОсозд.

Связь ЗАТРАЧИВАЕТСЯ удовлетворяет условиям правила 6, в соответствии с которым получаем три отношения:

1. Деталь (Номер, …)
2. Заказ-наряд (Номер, ФИОсозд, …)
3. Расход (НомерДетали, НомерНаряда) – добавили новую сущность Расход, которая содержит в себе два первичных ключа сущностей Деталь и Заказ-наряд.

Связь ИМЕЕТ АНАЛОГ удовлетворяет условиям правила 6, в соответствии с которым получаем три отношения:

1. Деталь (Номер, …)
2. Аналог (НомерДетали, НомерАналога) – добавили новую сущность Аналог, которая содержит в себе два первичных ключа деталей.

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

1. Клиент (ФИО, …)
2. Запись (idзаписи, ФИОклиент) – добавился неключевой атрибут ФИОклиент.

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

1. Транспортное средство (WIN-номер, ФИОдиагност, …)
2. Заказ-наряд (Номер, ФИОраб, WIN-номер …) – добавился неключевой атрибут WIN-номер.

Связь ПРЕДОСТАВЛЯЕТСЯ удовлетворяет условиям правила 6, в соответствии с которым получаем три отношения:

1. Заказ-наряд (Номер, ФИОсозд, WIN-номер …)
2. Услуга (Название, …)
3. УслугиЗН (НомерНаряда, НазваниеУслуги)

Создали новую сущность УслугиЗН, которая содержит оба первичных ключа сущностей Заказ-наряд и Услуга.

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

1. Клиент (ФИО, …)
2. Транспортное средство (WIN-номер, ФИОдиагност, ФИОклиента…) – добавился неключевой атрибут ФИОклиента.

Связь РАБОТАЕТ удовлетворяет условиям правила 6, в соответствии с которым получаем три отношения:

1. Работник (ФИО, …)
2. День (Дата, …)
3. Смена (ФИОраб, ДатаСмены)

Создали новую сущность Смена, которая содержит оба первичных ключа сущностей Работник и День.

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

Заказ-наряд (Номер, ФИОсозд, WIN-номер, СостояниеЗаказа, ДатаОформления, ДатаОкончания, Стоимость)

Транспортное средство (WIN-номер, ФИОдиагност, ФИОклиента, Марка, Модель)

Деталь (Номер, Производитель, Описание, Количество, ЕдИзм)

Клиент (ФИО, Дата рождения, пол, телефон)

Услуга (Название, Описание, Стоимость)

Запись (idЗаписи, ФИОклиент, Дата)

День (Дата)

Работник (ФИО, Дата рождения, должность, зарплата, стаж, телефон)

Аналог (НомерДетали, НомерАналога)

Расход (НомерДетали, НомерНаряда, Количество, ЕдИзм)

УслугиЗН (НомерНаряда, НазваниеУслуги)

Смена (ФИОраб, ДатаСмены)

1.4. Физическое проектирование

Физическое проектирование является третьим и последним этапом создания проекта базы данных, при выполнении которого разработчик принимает решения о способах реализации разрабатываемой базы данных. Во время предыдущего этапа проектирования была определена логическая структура базы данных (которая описывает отношения и ограничения в рассматриваемой прикладной области). 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.

Физическое проектирование состоит из трёх этапов:

1. Проектирование основных отношений

Определение каждого отношения включает:

* Имя отношения
* Список простых атрибутов
* Определение первичного ключа и альтернативных и внешних ключей
* Список производных атрибутов и описание способов их вычисления
* Определение требований ссылочной целостности для любых внешних ключей

Для каждого атрибута следует указать:

* Определение его домена, включающее тип данных, размерность внутреннего представления атрибута и любые требуемые ограничения на допустимые значения
* Значение атрибута, принимаемое по умолчанию
* Допустимость значения NULL

1. Разработка способов получения производных данных

Производный атрибут может храниться в БД или вычисляться каждый раз, когда в нём возникает необходимость.

1. Реализация ограничений предметной области

Существует несколько способов реализации ограничений:

* С использованием возможностей стандарта языка SQL
* С применением триггеров
* Непосредственно в самом приложении

В качестве СУБД был выбран MS SQL Server, а инструментом для работы с ним взята среда MS SQL Server Management Studio. Она представляет собой набор средств, для разработки сценариев на T-SQL и управления всеми компонентами SQL Server. Именно благодаря этому данная среда позволяет выполнять множество типовых операций, начиная с создания таблиц для баз данных, и управления созданными базами, заканчивая управлением безопасностью SQL Server.

Исходной информацией для этапа физического проектирования БД является логическая модель данных, состоящая из реляционной схемы БД, а также из ER-диаграмм или диаграмм отношений, созданных на предыдущем этапе проектирования. На основе этих отношений построим таблицы, которые лягут в основу базы данных.

В таблице “Работник” хранятся данные о работниках автосервиса.

Операторы создания этой и других таблиц представлены в приложении А, скриншоты заполнения таблиц данными – в приложении А.

Таблица 1

Требования к структуре таблицы «Работник»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Столбец** | **Тип данных** | **Нуль?** | **Ключ** | **Значение по умолчанию** | **Ограничение** | **Ссылка** |
| ФИО | Varchar (50) | Not null | Первичный |  |  |  |
| ДатаРождения | Date |  |  |  |  |  |
| Должность | Varchar (50) | Not null |  |  | “Механик”, “Менеджер”, “Директор” |  |
| Зарплата | Money |  |  |  |  |  |
| Стаж | Int | Not null |  | 1 | >0 |  |
| Телефон | Int | Not null |  |  |  |  |

Таблица 2

Требования к структуре таблицы «Клиент»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Столбец** | **Тип данных** | **Нуль?** | **Ключ** | **Значение по умолчанию** | **Ограничение** | **Ссылка** |
| ФИО | Varchar (50) | Not null | Первичный |  |  |  |
| ДатаРождения | Date | Not null |  |  |  |  |
| Телефон | Int | Not null |  |  |  |  |
| Пол | Char (1) |  |  | ‘м’ | ‘м’, ‘ж’ |  |

Таблица 3

Требования к структуре таблицы «ТранспортноеСредство»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Столбец** | **Тип данных** | **Нуль?** | **Ключ** | **Значение по умолчанию** | **Ограничение** | **Ссылка** |
| WINномер | Varchar (50) | Not null | Первичный |  |  |  |
| ФИОдиагност | Varchar (50) | Not null | Внешний |  |  | Работник(ФИО) |
| ФИОклиента | Varchar (50) | Not null | Внешний |  |  | Клиент(ФИО) |
| Марка | Varchar (50) |  |  |  |  |  |
| Модель | Varchar (50) |  |  |  |  |  |

Таблица 4

Требования к структуре таблицы «ЗаказНаряд»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Столбец** | **Тип данных** | **Нуль?** | **Ключ** | **Значение по умолчанию** | **Ограничение** | **Ссылка** |
| Номер | Int | Not null | Первичный |  |  |  |
| ФИОсозд | Varchar (50) | Not null | Внешний |  |  | Работник(ФИО) |
| WINномер | Varchar (50) | Not null | Внешний |  |  | ТранспортноеСредство(WINномер) |
| СостояниеЗаказа | Varchar (50) | Not null |  |  |  |  |
| Стоимость | Money | Not null |  |  | >0 |  |
| ДатаОформления | Date | Not null |  | Getdate() |  |  |
| ДатаОкончания | Date |  |  |  | >=ДатаОформления |  |

Таблица 5

Требования к структуре таблицы «Деталь»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Столбец** | **Тип данных** | **Нуль?** | **Ключ** | **Значение по умолчанию** | **Ограничение** | **Ссылка** |
| Номер | Varchar (50) | Not null | Первичный |  |  |  |
| Производитель | Varchar (50) |  |  |  |  |  |
| Описание | Varchar (50) | Not null |  |  |  |  |
| Количество | Int | Not null |  |  |  |  |
| ЕдИзм | Varchar (50) | Not null |  |  | ‘Литр’, ‘Штука’ |  |

Таблица 6

Требования к структуре таблицы «Услуга»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Столбец** | **Тип данных** | **Нуль?** | **Ключ** | **Значение по умолчанию** | **Ограничение** | **Ссылка** |
| Название | Varchar (50) | Not null | Первичный |  |  |  |
| Стоимость | Money | Not null |  |  |  |  |
| Описание | Varchar (50) | Not null | Первичный |  |  |  |

Таблица 7

Требования к структуре таблицы «Аналог»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Столбец** | **Тип данных** | **Нуль?** | **Ключ** | **Значение по умолчанию** | **Ограничение** | **Ссылка** |
| НомерДетали | Varchar (50) | Not null | Первичный, Внешний |  |  | Деталь(Номер) |
| НомерАналога | Varchar (50) | Not null | Первичный, Внешний |  |  | Деталь(Номер) |

Таблица 8

Требования к структуре таблицы «Запись»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Столбец** | **Тип данных** | **Нуль?** | **Ключ** | **Значение по умолчанию** | **Ограничение** | **Ссылка** |
| idЗаписи | Int | Not null | Первичный |  |  |  |
| ФИОклиента | Varchar (50) | Not null | Внешний |  |  | Клиент(ФИО) |
| Дата | Date | Not null |  | Getdate() |  |  |

Таблица 9

Требования к структуре таблицы «Расход»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Столбец** | **Тип данных** | **Нуль?** | **Ключ** | **Значение по умолчанию** | **Ограничение** | **Ссылка** |
| НомерДетали | Varchar (50) | Not null | Первичный, Внешний |  |  | Деталь(Номер) |
| НомерНаряда | Int | Not null | Первичный, Внешний |  |  | ЗаказНаряд(Номер) |
| Количество | Int | Not null |  |  | >0 |  |
| ЕдИзм | Varchar (50) | Not null |  |  | ‘Литр’, ‘Штука’ |  |

Таблица 10

Требования к структуре таблицы «УслугиЗН»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Столбец** | **Тип данных** | **Нуль?** | **Ключ** | **Значение по умолчанию** | **Ограничение** | **Ссылка** |
| НомерУслуги | Varchar (50) | Not null | Первичный, Внешний |  |  | Услуга(Название) |
| НомерНаряда | Int | Not null | Первичный, Внешний |  |  | ЗаказНаряд(Номер) |

Таблица 11

Требования к структуре таблицы «День»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Столбец** | **Тип данных** | **Нуль?** | **Ключ** | **Значение по умолчанию** | **Ограничение** | **Ссылка** |
| Дата | Date | Not null | Первичный |  |  |  |
| ЧасыРаботы | Varchar (50) |  |  | ‘8:00-20:00’ |  |  |

Таблица 12

Требования к структуре таблицы «Смена»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Столбец** | **Тип данных** | **Нуль?** | **Ключ** | **Значение по умолчанию** | **Ограничение** | **Ссылка** |
| ФИОраб | Varchar (50) | Not null | Первичный, Внешний |  |  | Работник(ФИО) |
| ДатаСмены | Varchar (50) | Not null | Первичный, Внешний |  |  | День(Дата) |

Схема БД, содержащая связанные таблицы, показана на скриншоте ниже (рис. 1.18).

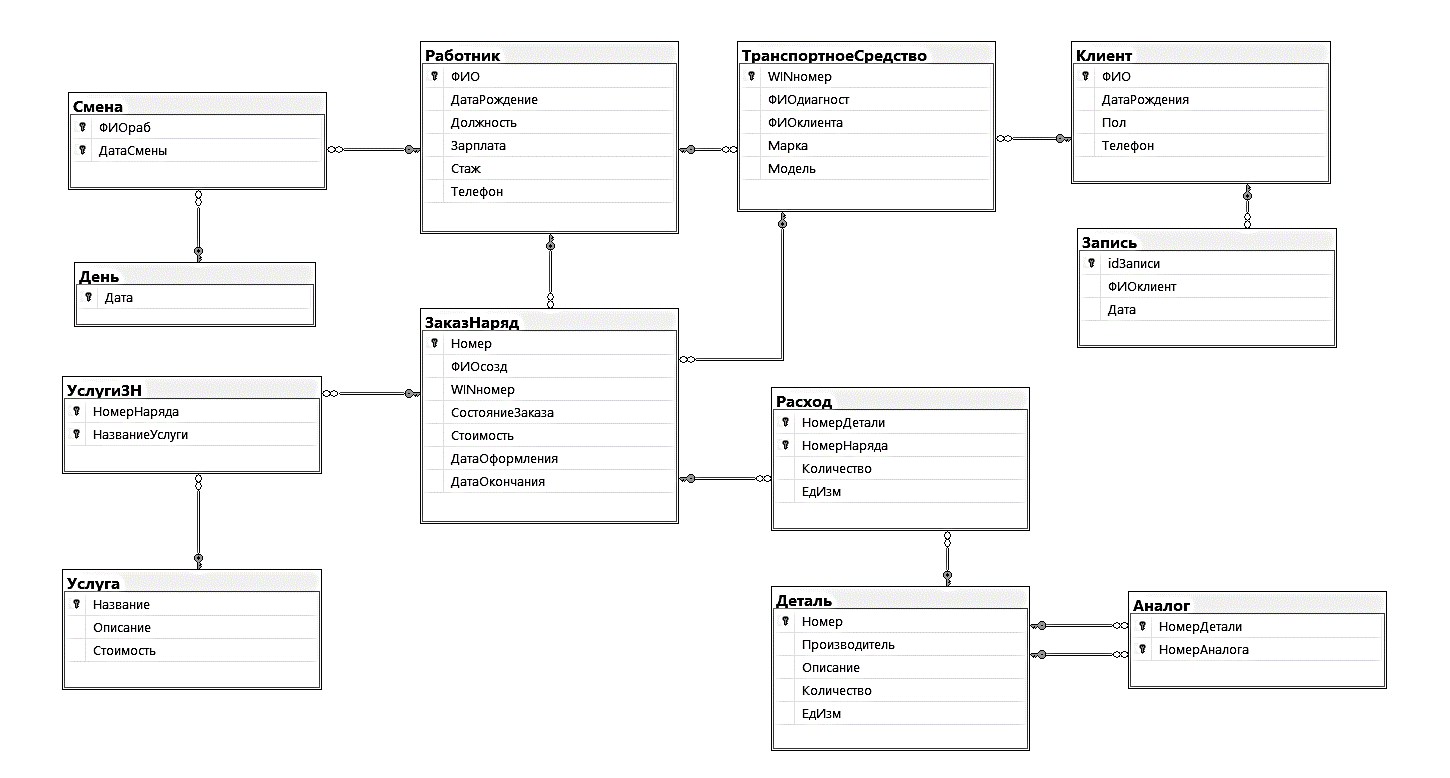


Рис. 1.18. Схема связанных таблиц БД

Функция вывода информации о данных сотрудника в личном кабинете реализована с помощью хранимой процедуры. Хранимая процедура “account\_data” (см. Б.1) осуществляет вывод Дату рождения, должность, зарплату, стаж, телефон, логин и пароль в соответствующих текс боксы. Входным параметров является первичный ключ таблицы “Работник”, т.е. ФИО сотрудника. Выходных параметров данная процедура не содержит.

Функцию изменения(обновления) данных рассмотрим на примере хранимой процедуры “account\_edit\_data” (см. Б.2). Данная процедура получает ФИО, дату рождения, телефон, логин и пароль работника и по первичному ключу изменяет данные в таблице работник на вновь поступившие. Выходных параметров в данной хранимой процедуре нет.

В рамках данного курсового проекта по аналогии с процедурами вывода данных из БД было создано большое количество хранимых процедур для изменения данных.

Ниже приведен список таких процедур:

* Admin\_edit\_detali (Изменение стоимости детали) (см. Б.59)
* Admin\_edit\_service (Изменение стоимости услуги) (см. Б.23)
* Admin\_edit\_sotr (Изменение данных сотрудника) (см. Б.21)
* manager\_edit\_client (Изменение данных клиента) (см. Б.12)
* Manager\_update\_zapisi (Изменение данных актуальной записи) (см. Б.59)
* Edit\_zakaz (Изменение данных заказ-наряда) (см. Б.30)
* Mechanik\_update\_count\_detali (Заказ нужных деталей, фактически изменение количества нужных деталей на складе) (см. Б.60)

Функция добавления данных в таблицу БД будет рассмотрена на примере хранимой процедуры “Add\_new\_client” (см. Б.13). На вход данная функция принимает четыре поля, ФИО клиента, дату его рождения, пол и телефон. Результатом выполнения функции является появление новой записи в таблице “Клиент”.

В рамках данного курсового проекта было создано большое количество аналогичных процедур.

Ниже приведен список таких процедур:

* add\_new\_zakaz (добавляет новый заказ-наряд и услугу “Диагностика” в таблицу “УслугиЗН”) (см. Б.26)
* add\_service\_to\_zakaz (Добавление услуги с “УслугиЗН”) (см. Б.7)
* admin\_add\_detali (Добавление новой запчасти в таблицу “Деталь”) (см. Б.27)
* Admin\_add\_new\_sotr (Добавление нового сотрудника в таблицу “Работник”) (см. Б.61)
* Admin\_add\_service (Добавление новой услуги в таблицу “Услуги”) (см. Б.22)
* Manager\_add\_new\_zapisi (Добавление новой записи клиента на посещение автосервиса) (см. Б.16)
* Mechanic\_new\_car (Добавление нового ТС в таблицу “ТранспортноеСредство”) (см. Б.8)

Отдельно нужно упомянуть о хранимой процедуре “Add\_detali\_to\_zakaz” (см. Б.25) потому как данная процедура совмещает в себе функционал внесения новых данных в таблицу Расход и обновляет соответствующие данные о деталях на складе.

Функция вывода табличной части таблицы будет рассмотрена на примере хранимой процедуры “Admin\_list\_of\_detali” (см. Б.24). Данная функция является одной из самых простых и содержит всего одну команду по выводу всех полей исходной таблицы в DataGriwView. Ни входных, ни выходных параметров хранимая процедура не имеет.

В рамках данного курсового проекта было создано большое количество аналогичных процедур.

Ниже приведен список таких процедур:

* Admin\_list\_of\_service (Выводит таблицу с данными об оказываемых услугах) (см. Б.10)
* Admin\_list\_of \_workers (Выводит таблицу с данными обо всех сотрудниках) (см. Б.62)
* list\_of\_days (Выводит таблицу с данными о сменах в личном кабинете сотрудника) (см. Б.34)
* list\_of\_detali (Выводит таблицу с данными о деталях на складе) (см. Б.24)
* list\_of\_service (Выводит таблицу с данными об оказываемых услугах) (см. Б.6)
* Manager\_list\_of\_clients (Выводит таблицу с данными о клиентах) (см. Б.11)
* Manager\_list\_of\_zakaz (Выводит сводную таблицу с данными о заказ-нарядах и информацией о транспортных средствах которые фигурируют в них) (см. Б.9)
* Manager\_list\_of\_zapisi (Выводит таблицу с данными об архивных записях) (см. Б.15)
* Manager\_list\_of\_zapisi\_actual (Выводит таблицу с данными об актуальных записях) (см. Б.14)
* Mechanic\_list\_of\_cars (Выводит таблицу с данными о транспортных средствах из таблицы “ТранспортноеСредство”) (см. Б.35)
* Mechanic\_list\_of\_orders (Выводит таблицу с данными о незавершенных заказ-нарядах) (см. Б.36)
* Архив (Выводит таблицу с данными завершённых заказ-нарядов) (см. Б.38)

Есть в базе данных несколько вспомогательных хранимых процедур, которые упрощают и делают работу приложения прогнозируемой и корректной.

Среди таких процедур есть три идентичных перед которыми поставлена задача выводить максимальный номер плюс один, т.е. новый номер для создания чего-то в БД. В случае хранимой процедуры “last\_num” (см. Б.3) она возвращает новый номер для создания заказ-наряда. А хранимая процедура “last\_num\_detali” (см. Б.32) возвращает новый номер для добавления новой детали в БД. И также как и в две предыдущие процедура “last\_num\_zapisi” (см. Б.33) возвращает новый номер для добавления записи клиента.

Также вспомогательной является процедура “diagnostic\_cost” (см. Б.4) которая имеет один выходной параметр, который возвращает стоимость услуги “Диагностика” в поле “Стоимость” при создании нового заказ-наряда.

Не подходящая под общую классификацию хранимая процедура “Manager\_delete\_zapisi” (см. Б.59) создана, чтобы удалять отмененную запись клиента на помещение автосервиса. Процедура имеет один входной параметр, первичных ключ таблицы “Запись”, т.е. номер записи.

Также по-своему уникальная хранимая процедура “filter\_wincode” (см. Б.31) созданная для реализации фильтрации списка транспортных средств по их WIN-номеру при создании нового заказ-наряда. Получает на вход часть WIN-номера на основе которого выводит список подходящий в “ComboBox”.

Глава 2. Описание функционирования БД

База данных была создана в приложении Microsoft SQL Server Management Studio 18. Приложение “Автосервис” реализовано с помощью .NET Framework на языке С# в среде разработки Visual Studio 2019.

Реализованное программное средство представляет из себя законченный информационный продукт, способный обеспечивать работу автосервиса, закрывая большую часть из числа функциональный потребностей всех сотрудников, работающий в автосервисе. Разработанная программа отвечает качествам адекватности, безопасности и качества.

2.1. Назначение и перечень функций базы данных

Прямое назначение базы данных и приложения автосервиса заключается в автоматизации процессов данного бизнеса.

Программа, разработанная по предварительно сформулированным требованиям проектирования БД включает в себя весь перечень описанных в ней функции. Данный список претерпел значительные изменения в ходе непосредственной реализации программного продукта, заметно увеличив существующий функционалу. Ниже приведен финальный список реализованных функций:

**Всем сотрудникам доступны следующие функции:**

* Регистрация
* Авторизация
* Доступ к личному кабинету

**Механику доступны следующие функции:**

* Просмотр и сортировка списка заказ-нарядов
* Поиск заказ-наряда по WIN-номеру ТС
* Изменение заказ-наряда
* Просмотр и сортировка архива заказ-нарядов
* Просмотр и сортировка списка транспортных средств
* Добавление нового транспортного средства
* Поиск транспортного средства по WIN-номеру
* Просмотр и сортировка списка деталей на складе
* Поиск необходимых деталей по описанию
* Заказ необходимых деталей на склад

**Менеджеру доступны следующие функции:**

* Просмотр и сортировка списка заказ-нарядов
* Многокритериальный поиск заказ-наряда
* Просмотр состава заказ-наряда
* Просмотр и сортировка списка оказываемых услуг
* Поиск услуги по названию
* Просмотр и сортировка списка клиентов
* Изменение данных клиента
* Добавление новых клиентов
* Просмотр списка записей
* Добавление, удаление и изменение записи

**Директору доступны следующие функции:**

* Просмотр и сортировка списка сотрудников
* Поиск сотрудника по ФИО
* Добавление нового сотрудника
* Просмотр и сортировка списка оказываемых услуг
* Поиск услуги по названию
* Добавление новой услуги
* Изменение стоимости услуги
* Просмотр и сортировка списка хранимых деталей
* Поиск детали по описанию
* Добавление новой детали
  1. Описание работы с базой данных

Форма “LoginForm.cs” (см. В.2) является стартовой и спроектирована для авторизации работника автосервиса.

Описание формы:

Форма содержит два TextBox для внесения логина и пароля пользователя. Также на форме расположены две кнопки. В левом верхнем углу находится кнопка, переводящая пользователя к регистрации новой учетной записи сотрудника.

При нажатии на нижнюю кнопку будет будет произведена проверка введенных данных и авторизация пользователя в его личное рабочее пространство.

Общий вид формы в режиме конструктора представлен на скриншоте ниже (рис. 2.1).

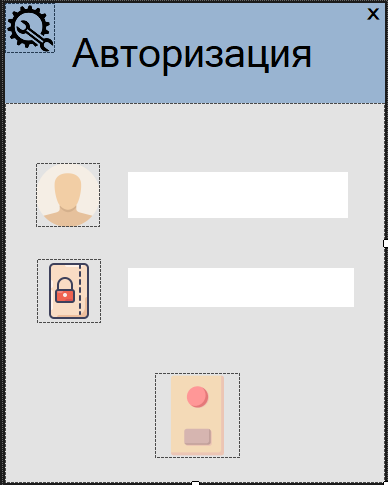


Рис. 2.1. Общий вид формы “Registration.cs”

Ниже приведен скриншот, демонстрирующий работу данной формы (рис. 2.2).

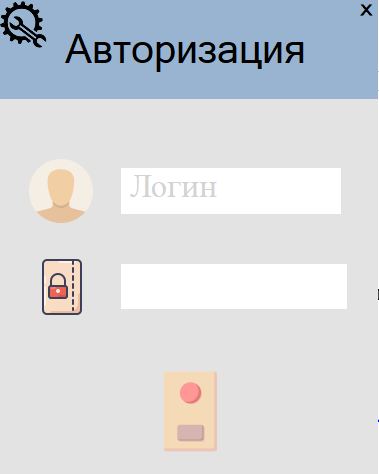


Рис. 2.2. Демонстрация работы “Registration.cs”

Нажав на кнопку в виде шестеренки на форме авторизации, мы попадаем на форму “Registration.cs” (см. В.1). Данная форма спроектирована для самостоятельной регистрации нового работника в системе.

Описание элементов:

На данной форме присутствует пять элементов TextBox в которые пользователь вводит соответствующие данные (ФИО, дату рождения, должность, логин и пароль).

Также на данной форме помимо красивого дизайна можно найти две функциональные кнопки. Первая кнопка в виде шестерни с ключом, расположенная в левом верхнем углу открывает форму “LoginForm.cs” для авторизации существующих пользователей. Вторая кнопка расположенная по внизу по центру формы предназначена для вызова SQL-запроса на добавление сотрудника в БД.

Форма также предполагает три проверки вводимых пользователем данных: проверка существования пользователя, проверка корректности должности, проверка даты рождения.

Общий вид формы в режиме конструктора представлен на скриншоте ниже (рис. 2.3).

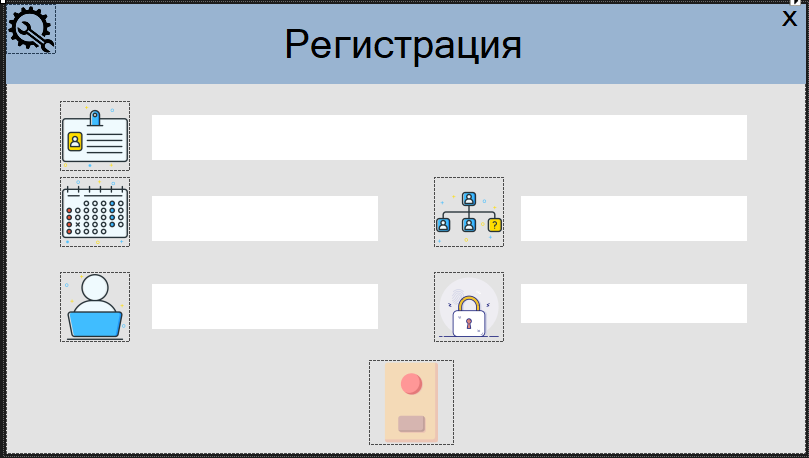


Рис. 2.3. Общий вид формы “Registration.cs”

Ниже приведен скриншот, демонстрирующий работу данной формы (рис. 2.4).

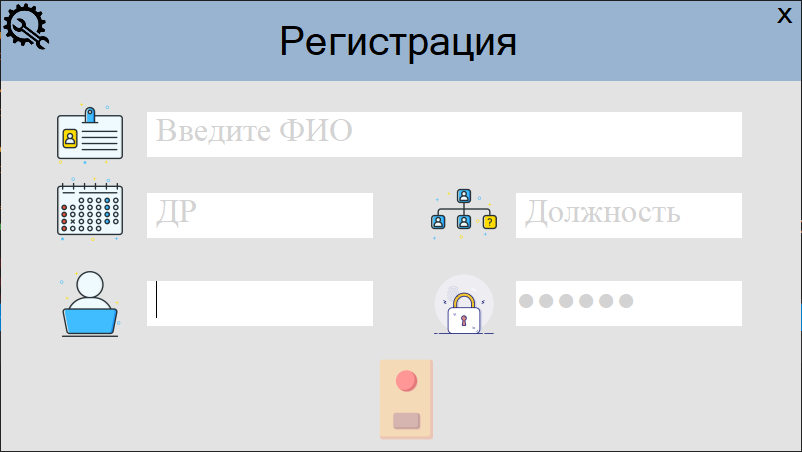


Рис. 2.4. Демонстрация работы “Registration.cs”

Авторизовавшись за сотрудника в должности “Механик” мы попадаем на форму Mechanic.cs (см. В.20). Данная является ключевой с точки зрения бизнес-процессов, протекающий в автосервисе.

Описание формы:

Данная форма содержит множество функциональных элементов, стоит отменить, что все они структурированы по соответствующим вкладкам.

Первая вкладка “Заказ-наряды” содержит DataDridView, выводящий список актуальных заказ-нарядов авторизированного механика.

Также на данной вкладке реализован блок сортировки списка заказ-нарядов, расположенный справа. Под таблицей расположилось текстовое поле для поиска конкретного заказ-наряда по WIN-номера транспортного средства, фигурирующего в нем. Справа под блоком сортировки находятся две кнопки, “Найти” и “Показать все” приводящие фильтрацию в действие.

Отдельно стоит упомянуть о двух кнопках, расположенных в верхней части формы. Крайняя левая кнопка вернет пользователя на этап авторизации. Кнопка расположенная немного правее откроет личный кабинет авторизированного пользователя, форма “Account\_form.cs”.

Также стоит упомянуть о кнопке “добавить”, которая, согласно своему названию, откроет новую форму “Mechanic\_add\_form.cs” для создания нового заказ-наряда.

Общий вид вкладки “Заказ-наряды” в режиме конструктора представлен на скриншоте ниже (рис. 2.5).

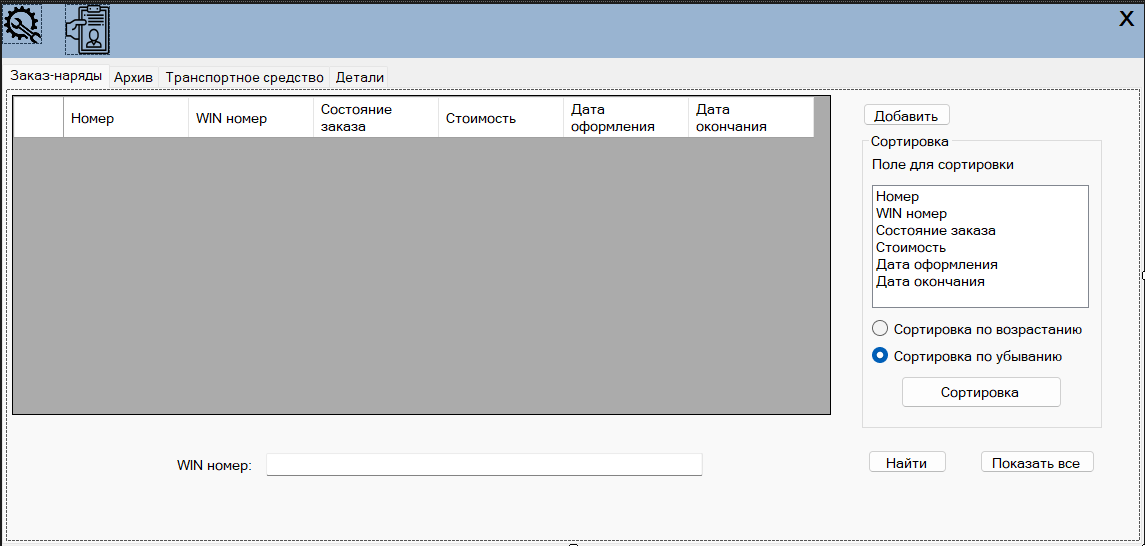


Рис. 2.5. Форма “Mechanic.cs” вкладка заказ-наряды

Заполнение списка заказ-нарядов происходит благодаря хранимой функции “Mechanic\_list\_of\_orders”(см. Б.36).

Ниже приведен скриншот, демонстрирующий работу данной формы (рис. 2.6).

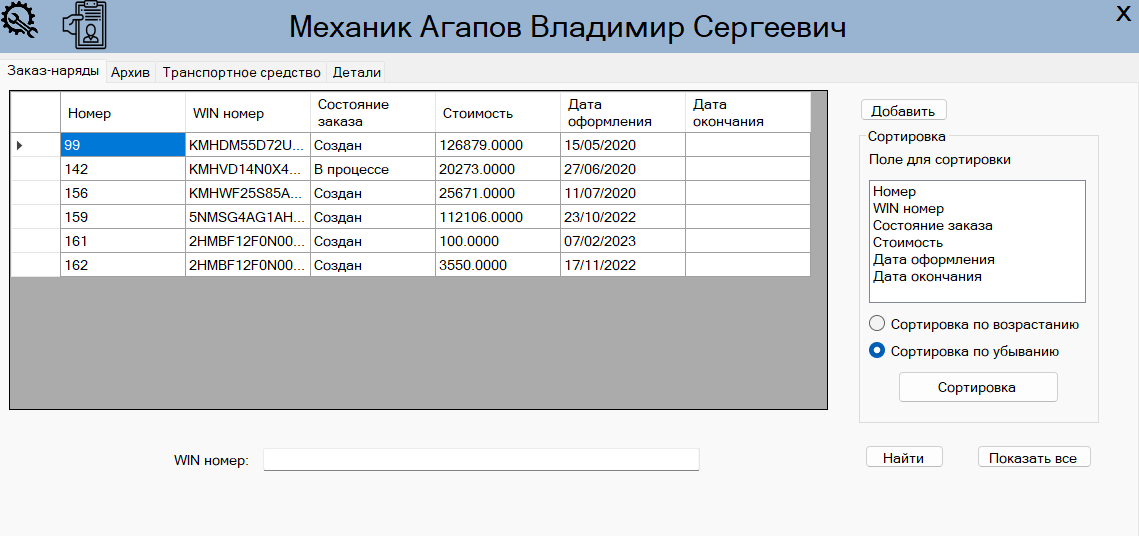


Рис. 2.6. Демонстрация работы “Mechanic.cs” вкладка заказ-наряды

При нажатии на пнопку “Добавить” будет вызвана форма “Mechanic\_add\_form”(см. В.21), созданная для добавления новых заказ-нарядов.

Общий вид формы “Mechanic\_add\_form” приведен ниже на скриншоте (рис. 2.7).

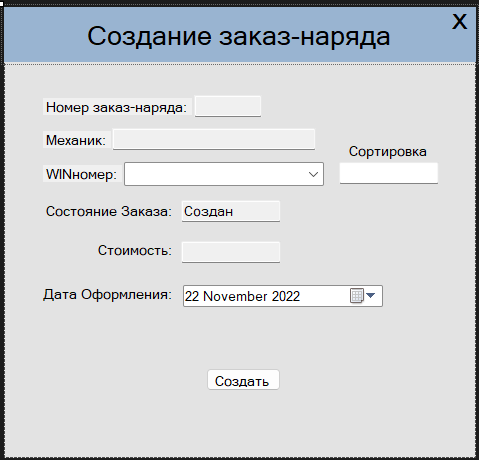


Рис. 2.7. Форма “Mechanic\_add\_form”

Серые текстовые поля заполняются при загрузке данной формы и не могут быть изменены. Данные о номере заказа выводятся с помощью специальной хранимой процедуры "last\_num" (см. Б.3), которая имеет один выходной параметр для этого.

Также для формирования первоначальной стоимости заказа была создана хранимая процедура “diagnostic\_cost”(см. Б.4), которая заполняет поле “Стоимость” стоимость услуги “Диагностика”.

Справа от поля WIN-номера располагается поле фильтрации списка WIN-номеров для удобства поиска нужного транспортного средства при создании нового заказа.

При нажатии кнопки создать, будет вызвана хранимая процедура “add\_new\_zakaz” (см. Б.26), созданная для добавления новых записей в таблице “Заказ-наряд”. Помимо непосредственного добавления нового заказ наряда с указанными на форме данными также будет добавлена услуга “Диагностика” в таблицу “УслугиЗН”.

В случае введения даты из прошлого, либо если указанный WIN-номер не фигурирует в базе данных сработает триггер “Check\_zakaz\_insert” (см. В.39), который отменит проведение транзакции и всплывет сообщение об ошибке.

Если все данные были введены правильно программа также выведет всплывающие окно с сообщением об успешном выполнении операции.

Ниже приведен скриншот, на котором показана работа формы (рис. 2.8).

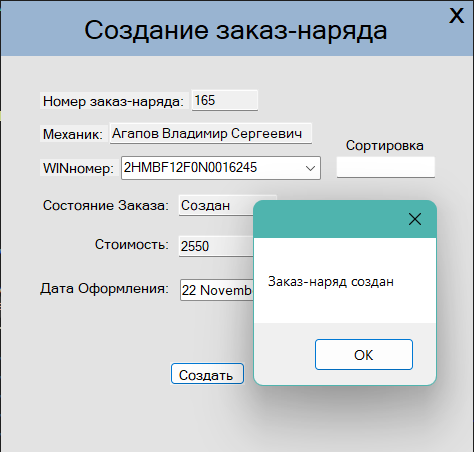


Рис. 2.8. Демонстрация работы “Mechanic\_add\_form”

Если на вкладке заказ-наряд (рис. 2.6) дважды щелкнуть по заказ-наряду откроется форма изменения данного заказ-наряда “Mechanik\_edit\_form” (см. В.23).

На форме можно выделить две ключевые функциональные зоны:

Левая часть формы наполнена текстовыми полями, относящимися непосредственно к данным заказ-наряда (ФИО механика, состояние, стоимость и так далее).

В правой части расположен блок, состоящий из двух таблиц, выводящий данные о услугах оказанных в рамках данного заказ-наряда и материалах, затраченных при ремонте транспортного средства.

Общий вид формы “Mechanic\_edit\_form” приведен ниже на скриншоте (рис. 2.9).

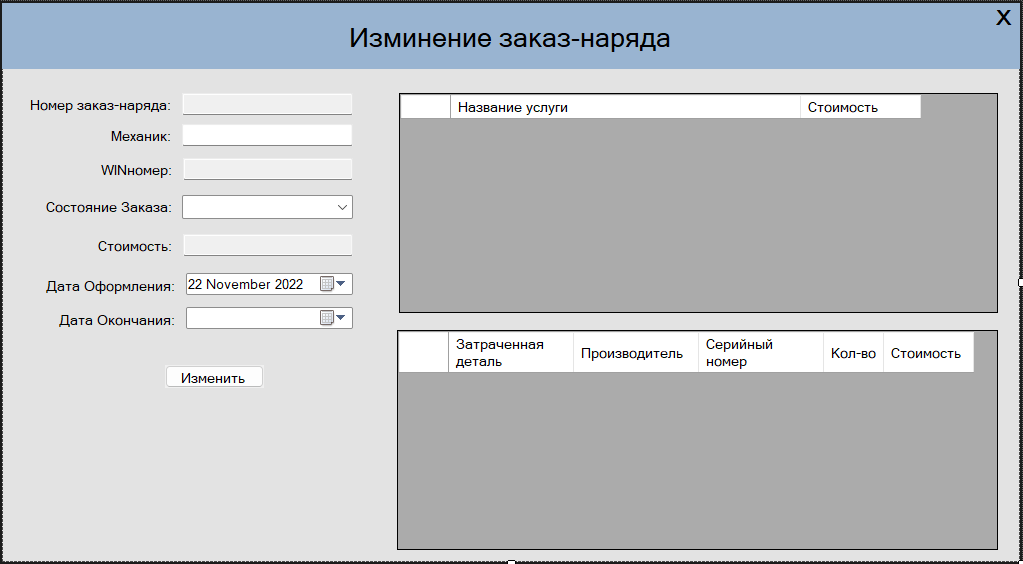


Рис. 2.9. Форма “Mechanic\_edit\_form”

Заполнение таблицы с услугами происходит благодаря хранимой процедуре “list\_of\_service” (см. Б.6), принимающая на вход номер заказ-наряда и возвращающая список услуг, оказанных в рамках данного заказ-наряда.

Заполнение таблицы деталей происходит аналогичным образом благодаря хранимой процедуре “list\_of\_detali” (см. Б.5).

После внесения изменений в заказ и нажатия кнопки “Изменить” будет вызвана хранимая процедура “Edit\_zakaz” (см. Б.30), которая применит все изменения на стороне БД.

Ниже приведен скриншот, на котором показана работа данной формы (рис. 2.10).

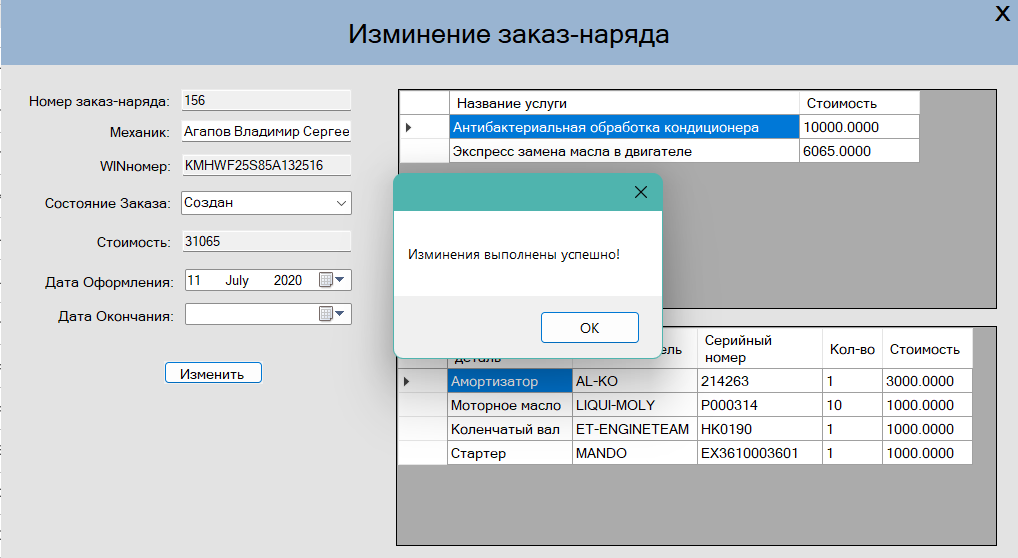


Рис. 2.10. Демонстрация работы “Mechanic\_edit\_form”

В случае, если была изменена изначальная дата оформления заказа, либо если имя механика не числится в списке сотрудников, либо если состояние заказа указано неверно при попытке изменить данные заказ-наряда сработает триггер “Check\_zakaz\_update” (см. Б.40), который отменит данную операцию и выдаст пользователю сообщение об ошибке.

Если сотрудник хочет добавить услугу в заказ наряд нужно нажать дважды по таблице с услугами и откроется новое окно формы “Add\_service\_form” (см. В.6). Данная форма является очень простой и содержит всего один выдающий список оказываемых услуг. Выбрав нужную услугу и нажав на кнопку добавить будет вызвана, хранимая процедура "Add\_service\_to\_zakaz" (см. Б.7) которая проведен операцию добавления новой услуги в заказ-наряд.

Ниже приведен скриншот, на котором показана работа данной формы (рис. 2.11).

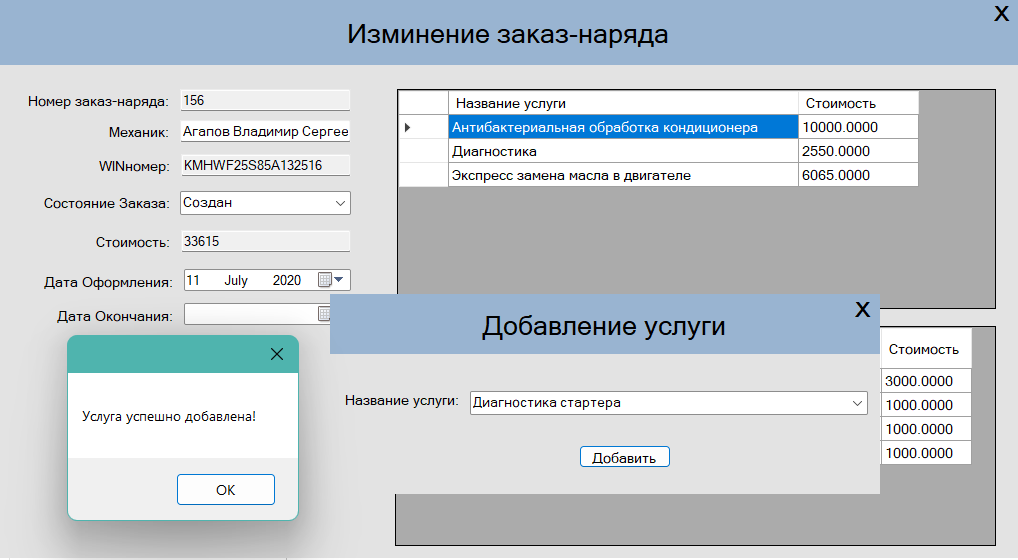


Рис. 2.11. Демонстрация работы “Add\_service\_form”

В случае, если такая услуга уже была оказана в рамках данного заказа будет выдано сообщение об ошибке и вызван триггер “Check\_service” (см. Б42), который отменит проведение операции.

Ниже приведен скриншот, показывающий работы данного триггера (рис. 2.12).

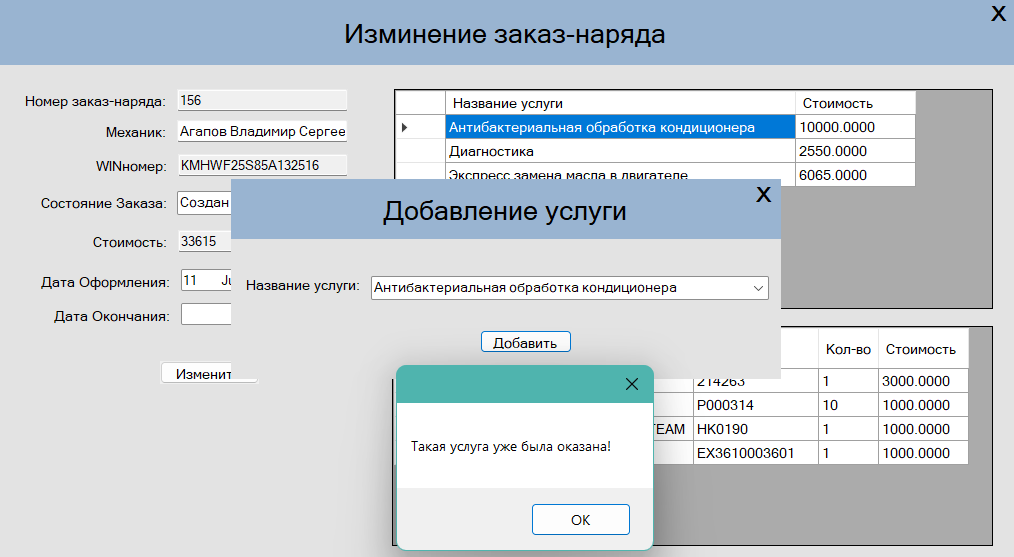


Рис. 2.12. Демонстрация работы триггера “Check\_service”

Аналогично если сотрудник хочет добавить деталь в заказ наряд нужно нажать дважды по таблице с детали и откроется новое окно формы “Add\_detali\_form”(см. Б.4). Данная форма чуть сложней предыдущей и содержит три выпадающих списка, которые заполняются согласно выбранному типу детали и производителю. Выбрав нужную услугу и нажав на кнопку добавить будет вызвана, хранимая процедура "Add\_detali\_to\_zakaz" (см. Б.25) которая проведен операцию добавления новой услуги в заказ-наряд.

Ниже приведен скриншот, на котором показана работа данной формы (рис. 2.13).

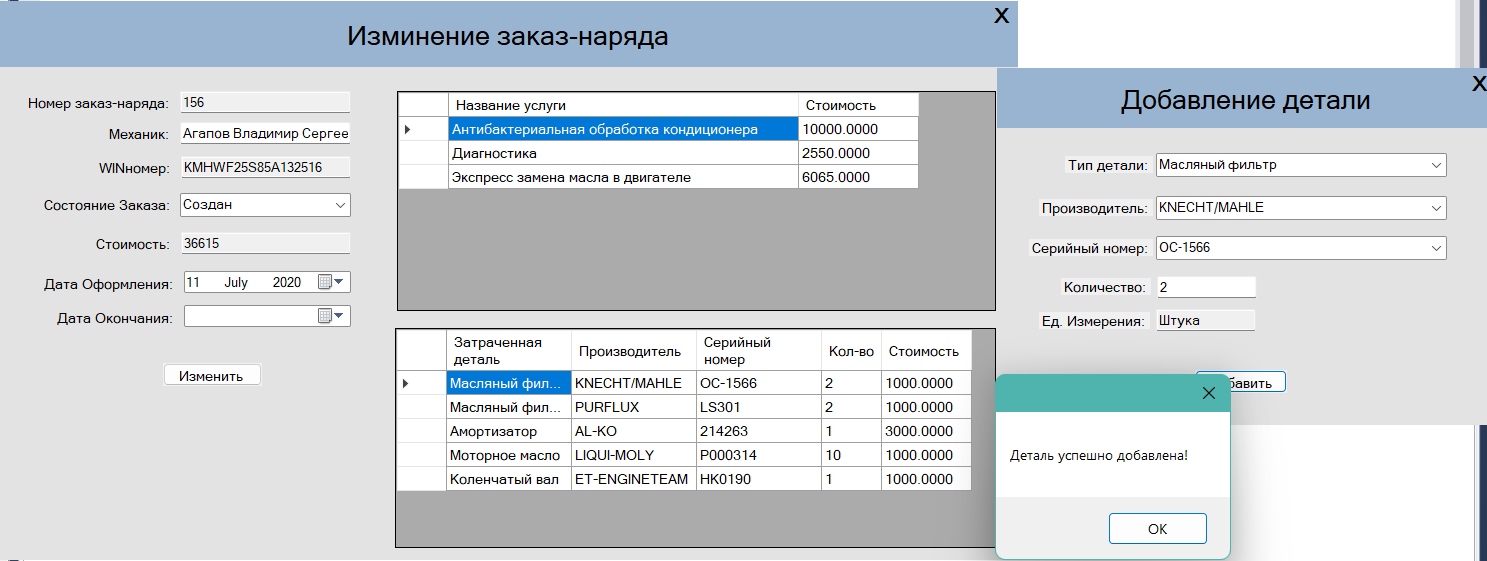


Рис. 2.13. Демонстрация работы “Add\_detali\_form”

В случае, если было заказано количество деталей, превышающее запас на складе, либо данная деталь уже фигурирует в рамках данного заказ-наряда, либо, если введенное количество отрицательно транзакция будет отменена вызванным триггером “Check\_count\_detali” (см. Б.43).

Ниже приведен скриншот, показывающий работы данного триггера (рис. 2.14).

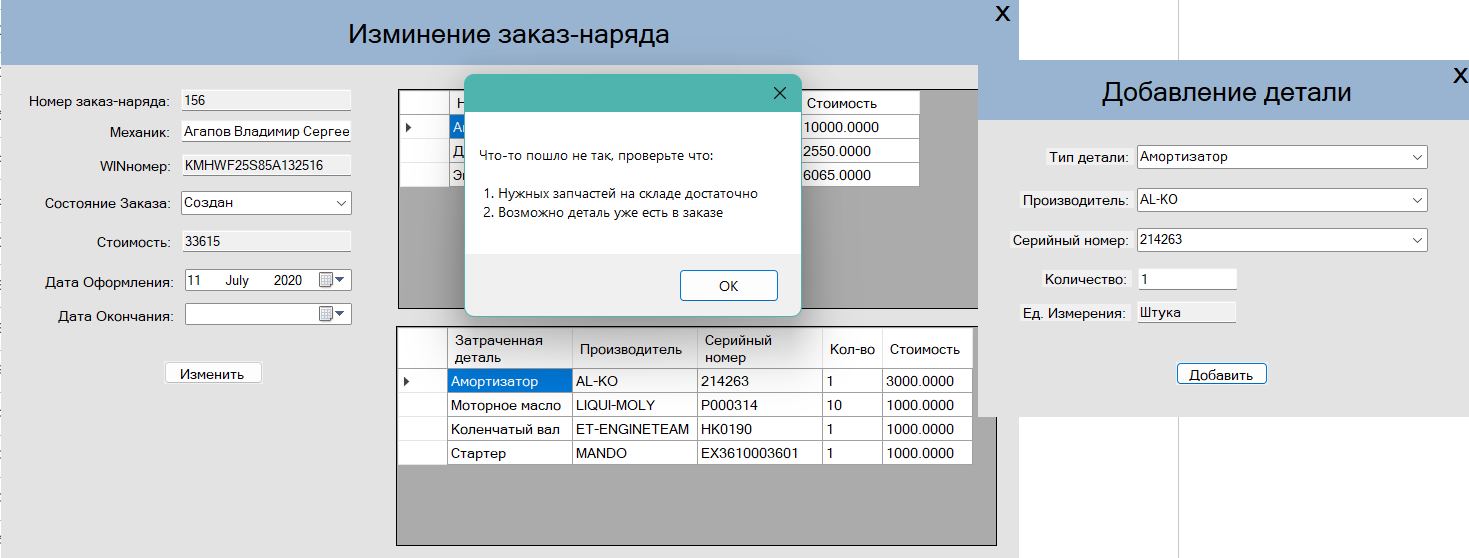


Рис. 2.14. Демонстрация работы триггера “Check\_service”

Закончили работу над заказ-нарядом, возвращаемся на вкладку заказ-наряды, посмотрим, как работает фильтрация по WIN-номеру. Ниже приведен скриншот, на котором продемонстрирована работа данной опции (рис. 2.15).

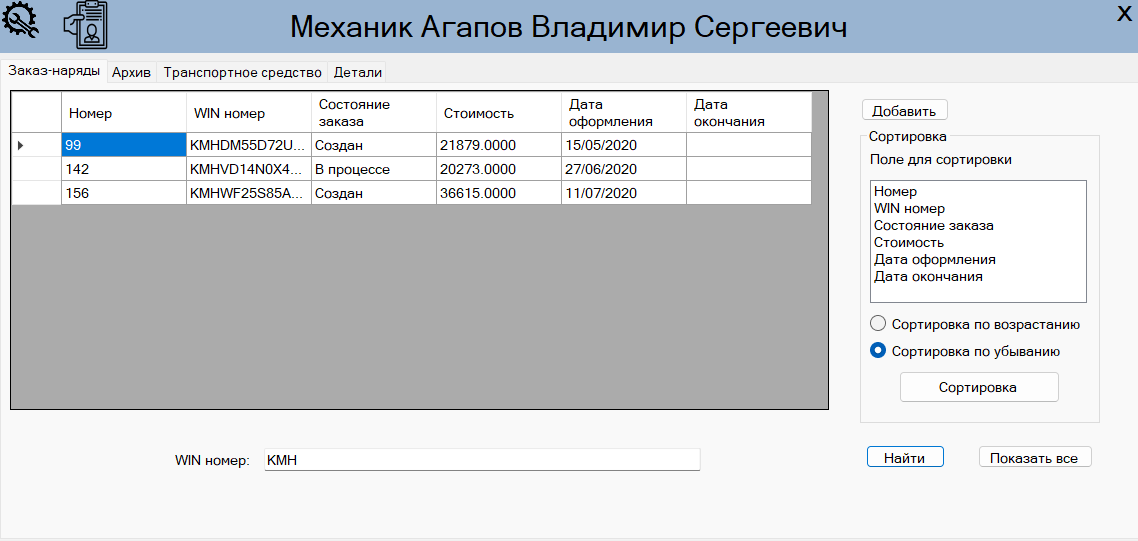


Рис. 2.15. Демонстрация работы фильтрации по WIN-номеру

Нажав на вторую вкладку “Архив”, работник увидит список архивных заказ-нарядов. Концепция данной формы полностью копирует устройство предыдущей вкладки “Заказ-наряды”, предоставляя возможность работнику просмотреть список всех заказ-нарядов, которые он завершил.

Ниже приведен скриншот, на котором показана работа данной формы (рис. 2.16).

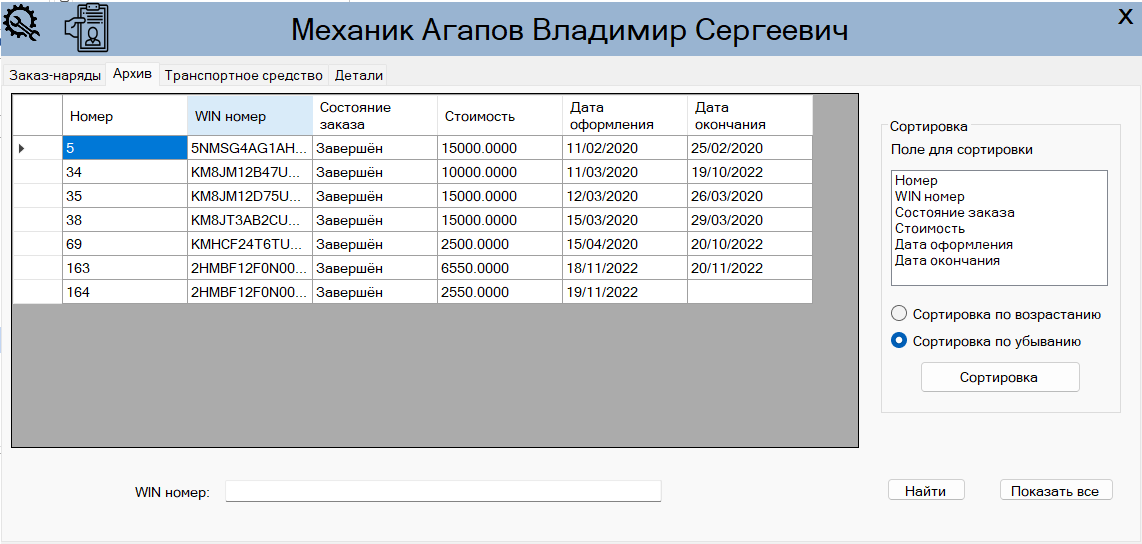


Рис. 2.16. Демонстрация работы “Mechanic.cs” вкладка архив

Нажав на третью вкладку “Транспортное средство”, откроется знакомый нам концепт представления информации. На данной вкладке также как и на двух предыдущих есть список, блок сортировки и поиска.

Ниже приведен скриншот, на котором показана работа данной формы (рис. 2.17).

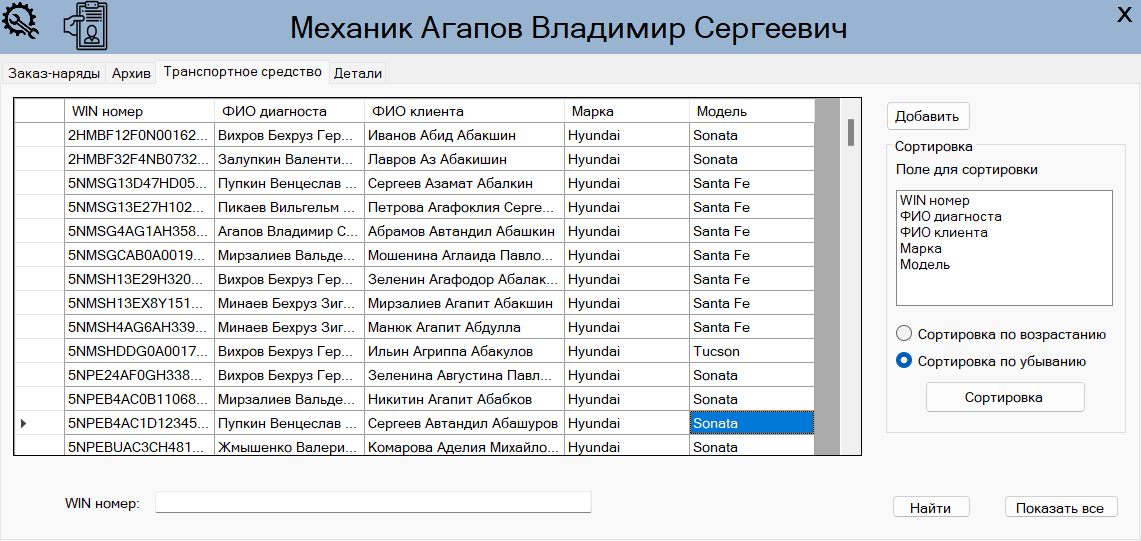


Рис. 2.17. Демонстрация работы “Mechanic.cs” вкладка транспортное средство

Нажав на кнопку добавить, откроется форма добавления нового транспортного средства “Mechanik\_new\_car\_form” (см. В.24).

Данная форма построена по образцу и подобию любой другой формы добавления чего-либо в таблицы БД. Однако стоит отметить интересную особенность данной формы. Чаще всего новое транспортное средство нужно добавлять для новых клиентов автосервиса, поэтому была добавлена специальная клетка “Новый клиент” для фильтрации клиентов по наличии автомобилей в базе автосервиса.

Общий вид формы “Mechanik\_new\_car\_form” приведен ниже на скриншоте (рис. 2.18.).

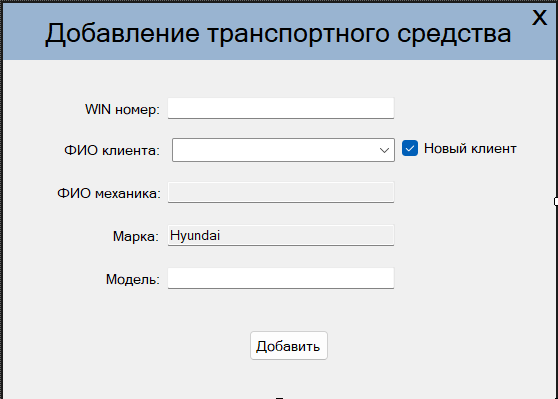


Рис. 2.18. Форма “Mechanik\_new\_car\_form”

По щелчку кнопки добавить будет вызвана хранимая процедура “Mechanic\_new\_car”(см. Б.8). Если все данные были указаны правильно программа отработает корректно и выведет об этом сообщение (рис. 2.19).

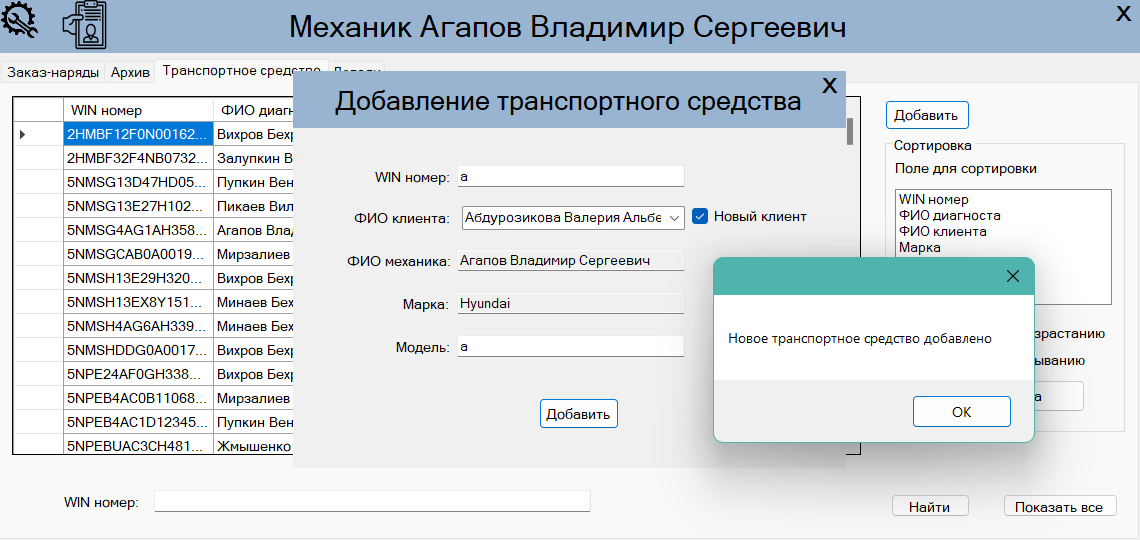


Рис. 2.19. Корректная работа формы “Mechanik\_new\_car\_form”

В случае, если данные были введены с ошибками сработает триггер “Check\_win” (см. Б.43) (рис. 2.20).

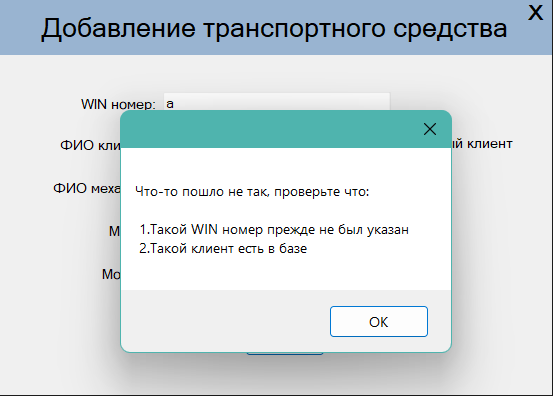


Рис. 2.20. Демонстрация работы триггера “ Check\_win”

На этом основной функционал механика закончился, однако важно отметить наличие общей для всех работников функции, а именно личный кабинет. Если пользователь нажнет на кнопку с человечком внутри планшета, то откроется форма “Account\_form”(см. В.3). На данной форме расположено две вкладки. На первой вкладе “Смены” (рис. 2.21) расположен DataGridView в котором выводятся все смены данного сотрудника благодаря хранимой процедуре “account\_data” (см. Б.1).

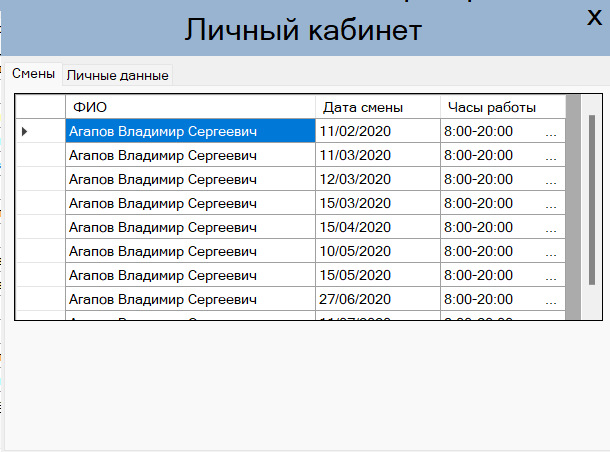


Рис. 2.21. Демонстрация работы “Account\_form” вкладка смены

На второй вкладке “Личные данные” расположены соответствующие поля с данными сотрудника (рис.2.22).

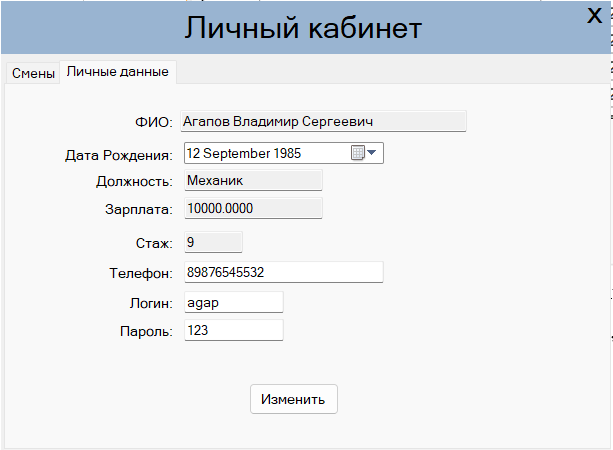


Рис. 2.22. Демонстрация работы “Account\_form” вкладка личные данные

Сотрудник может изменить дату своего рождения, телефон, логин и пароль. Если все данные были указаны корректно хранимая процедура "account\_edit\_data" (см. Б.2) применит изменения и внесет соответствующие правки в БД (рис. 2.23).

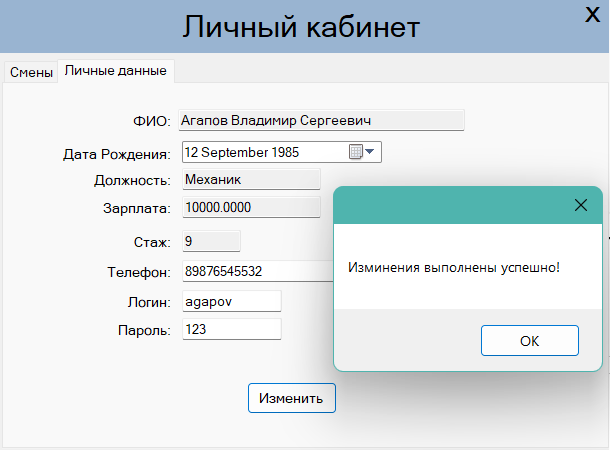


Рис. 2.23. Успешное изменение личных данных сотрудника

В случае, если какое-то из данных перечисленных выше было указано неверно, транзакция будет прервана триггером “Check\_sotr\_update”(см. Б.44) (рис. 2.24).

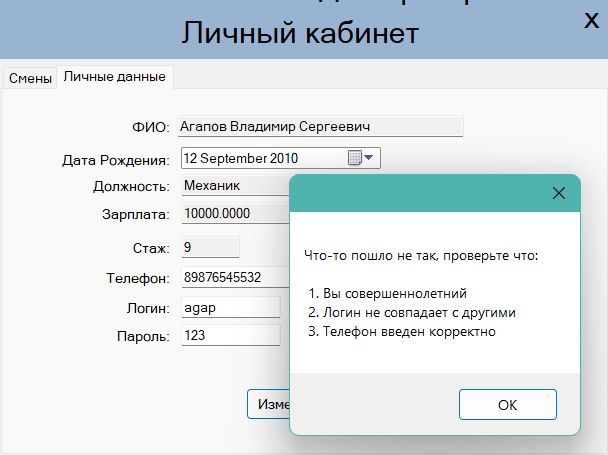


Рис. 2.24. Демонстрация работы триггера “ Check\_sotr\_update”

На данном весь спроектированный для механика функционал исчерпан. В случае, если на этапе авторизации пользователь зайдет в аккаунт менеджера, то откроется форма “Manager” (см. В.14).

Первая вкладка “Список заказ-нарядов” сильно перекликается с аналогичной вкладкой в рабочей среде механика, однако для менеджера наполнение списка было расширено, добавлена информация о транспортном средстве, участвующем в каждом заказ-наряде. Вывод данных в DataGridView реализовано по средствам хранимой процедуры “Manager\_list\_of\_zakaz”(см. Б.9) (рис. 2. 25).

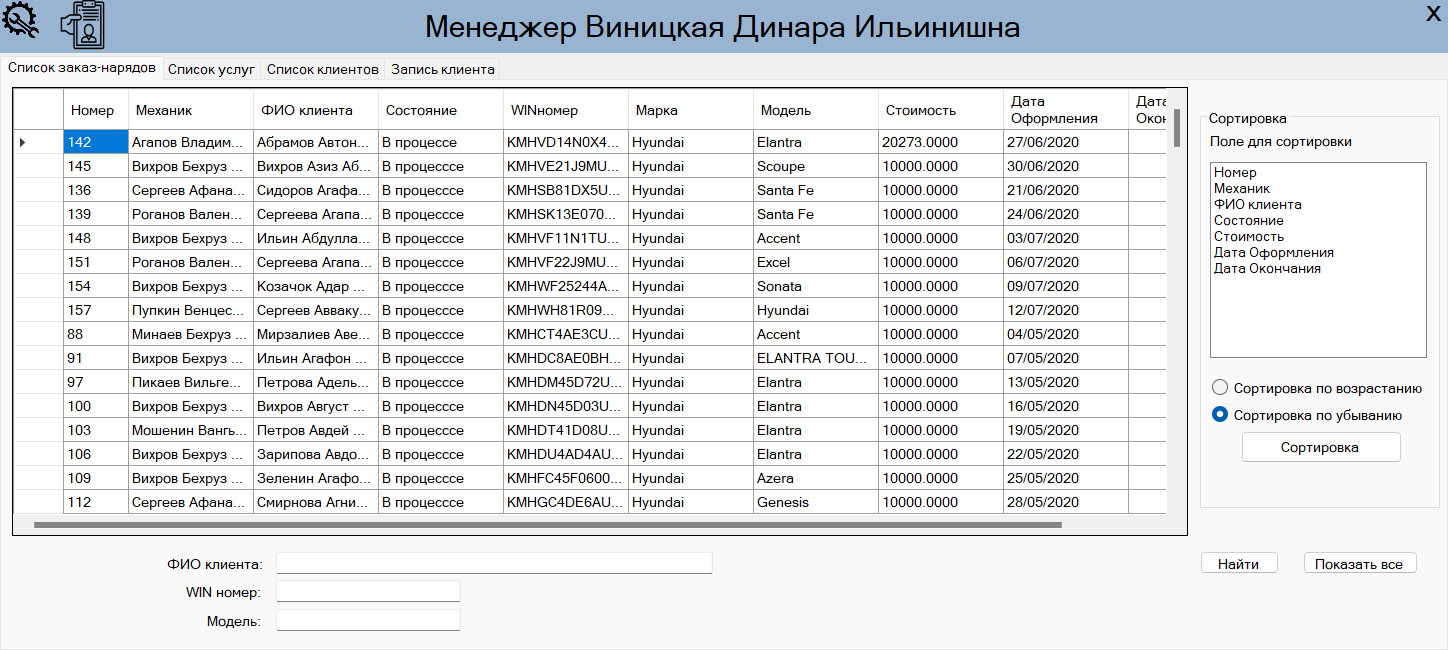


Рис. 2.25. Демонстрация работы “Manager” вкладка список заказ-нарядов

Также для удобства работы менеджера и обеспечении наилучшего качества обслуживания клиента был реализован многокритериальный поиск заказ-наряда, работающий аналогично обычному поиску, рассмотренному ранее. Однако многокритериальный поиск предполагает более детальный поиск конкретного заказ наряда среди общего количества по трем параметрам, ФИО клиента, WIN-номеру транспортного средства и модели машины (рис. 2.26).

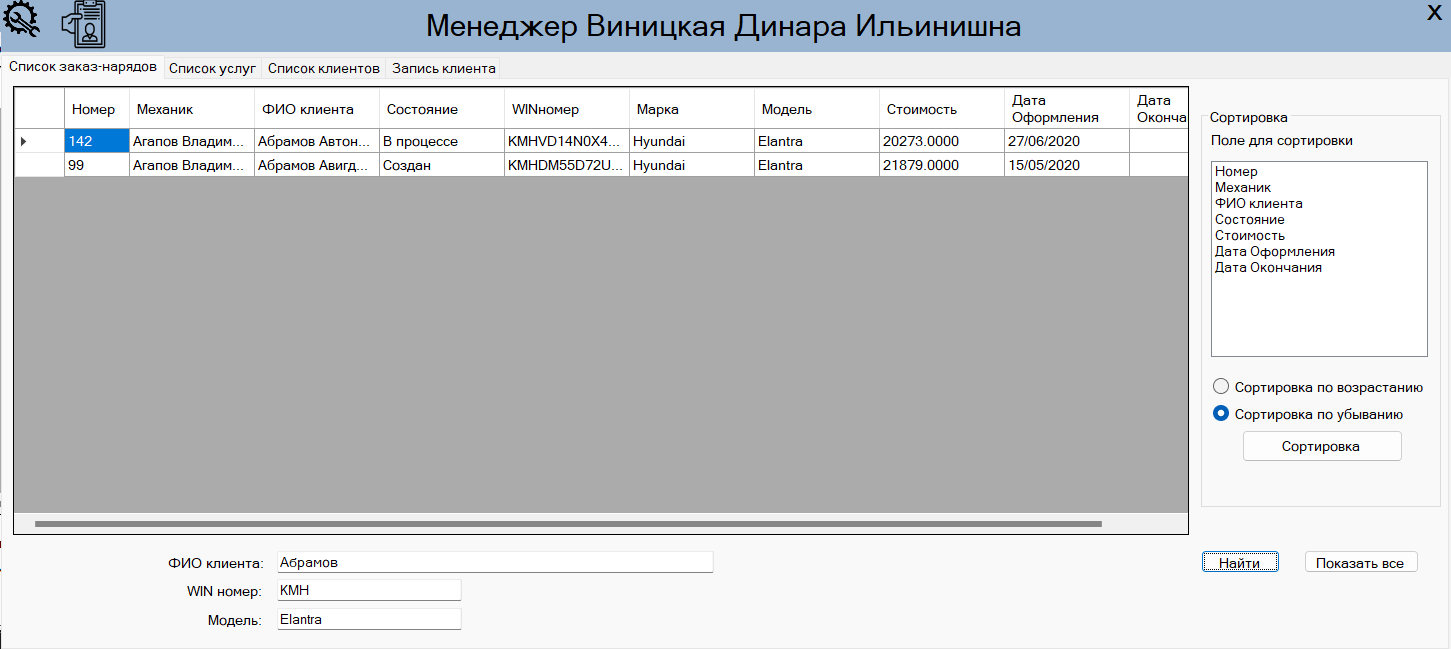


Рис. 2.26. Демонстрация работы многокретериального поиска

Менеджер может дважды щелкнуть по любому заказ-наряду из списка после чего будет вызвана форма “Manager\_zakaz\_details\_form”(см. В.19). За заполнение таблиц оказанных услуг и затраченных запчастей отвечают соответсвенно хранимые процедуры “list\_of\_detali” (см. Б.5) и “list\_of\_service” (см. Б.6) (рис. 2.27).

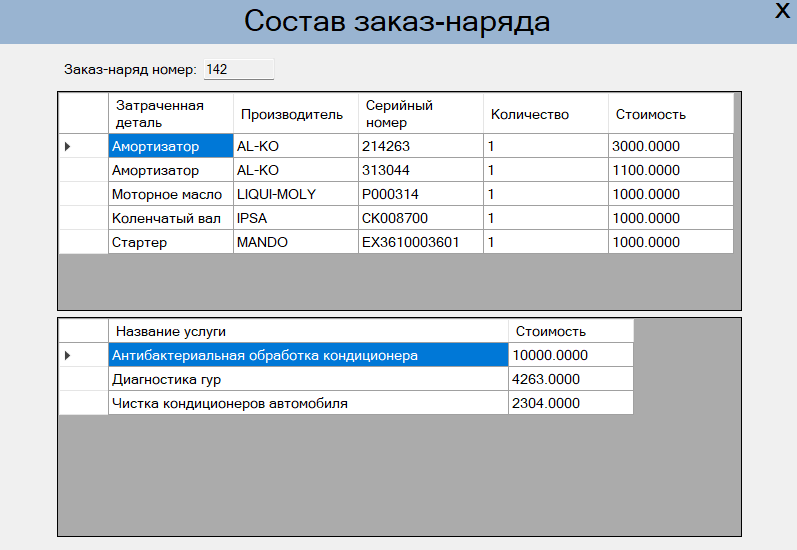


Рис. 2.27. Демонстрация работы формы “Manager\_zakaz\_details\_form”

На второй вкладке “Список услуг” взору менеджера открывается список услуг, предоставляемых автосервисом. Вывод данных реализован с помощью хранимой процедуры “Admin\_list\_of\_service” (см. Б.10) (рис 2.28).

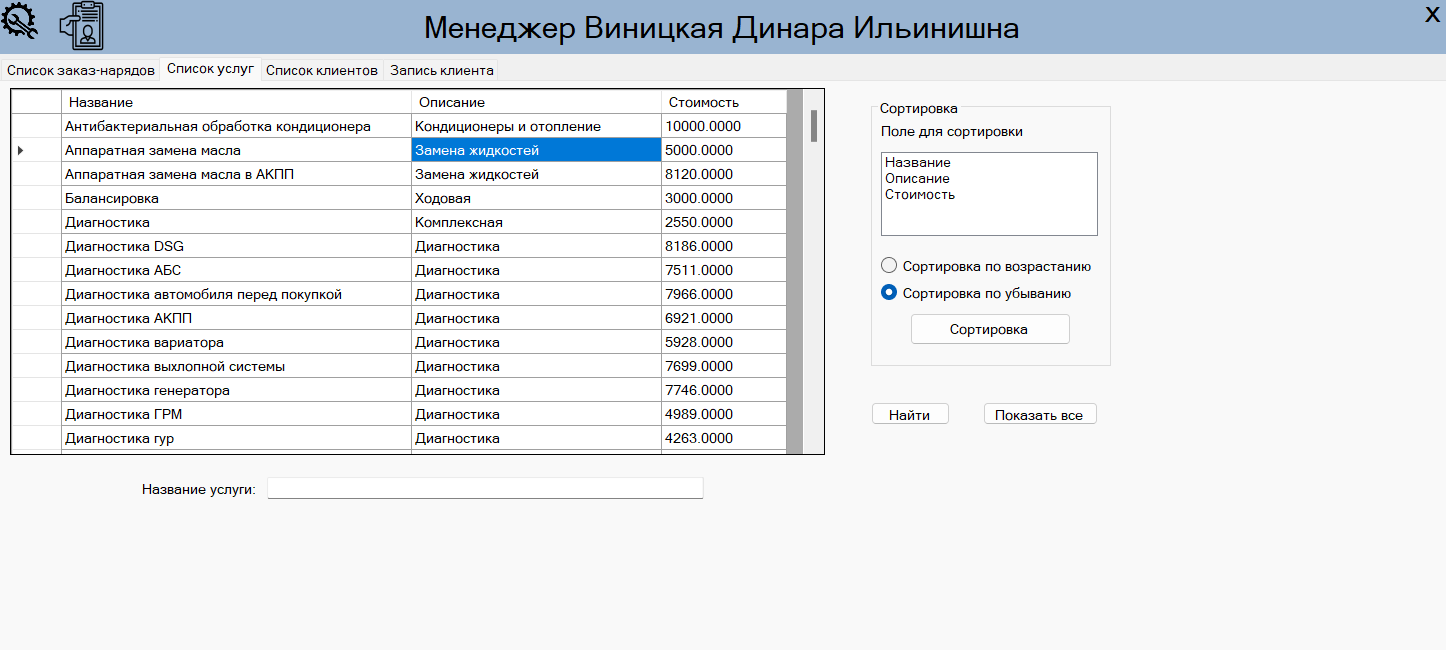


Рис. 2.28. Демонстрация работы формы “Manager” вкладка список услуг

На третьей вкладке “Список клиентов” менеджер увидит таблицу с данными всех клиентов автосервиса. Вывод данных реализован с помощью хранимой процедуры “Manager\_list\_of\_clients” (см. Б.11) (рис. 2.29).

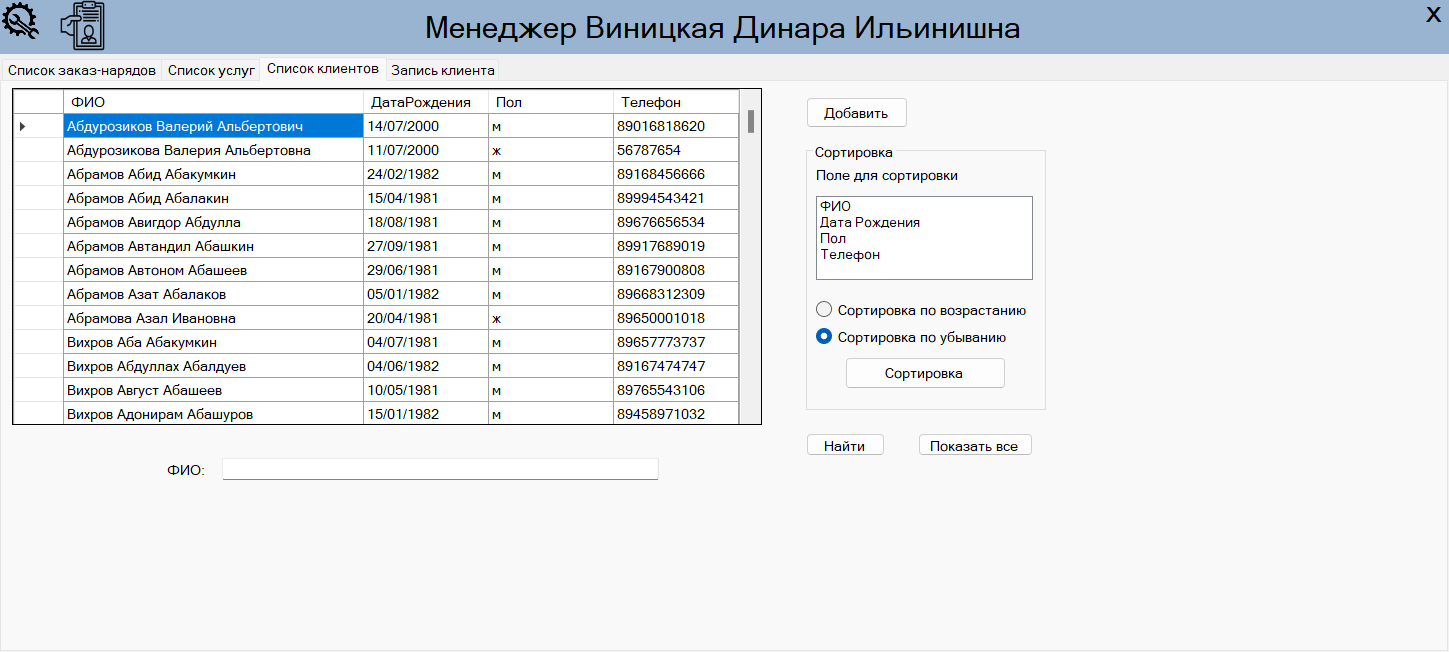


Рис. 2.29. Демонстрация работы формы “Manager” вкладка список клиентов

Дважды щелкнув по любому из клиентов в списке, менеджер откроет форму для изменения данных клиента “Manager\_edit\_client\_form” (см. В.17). В случае если изменения данных были проведены корректно, после нажатия кнопки изменить данные будет изменены благодаря хранимой процедуре “manager\_edit\_client” (см. Б.12) (рис. 2.30).

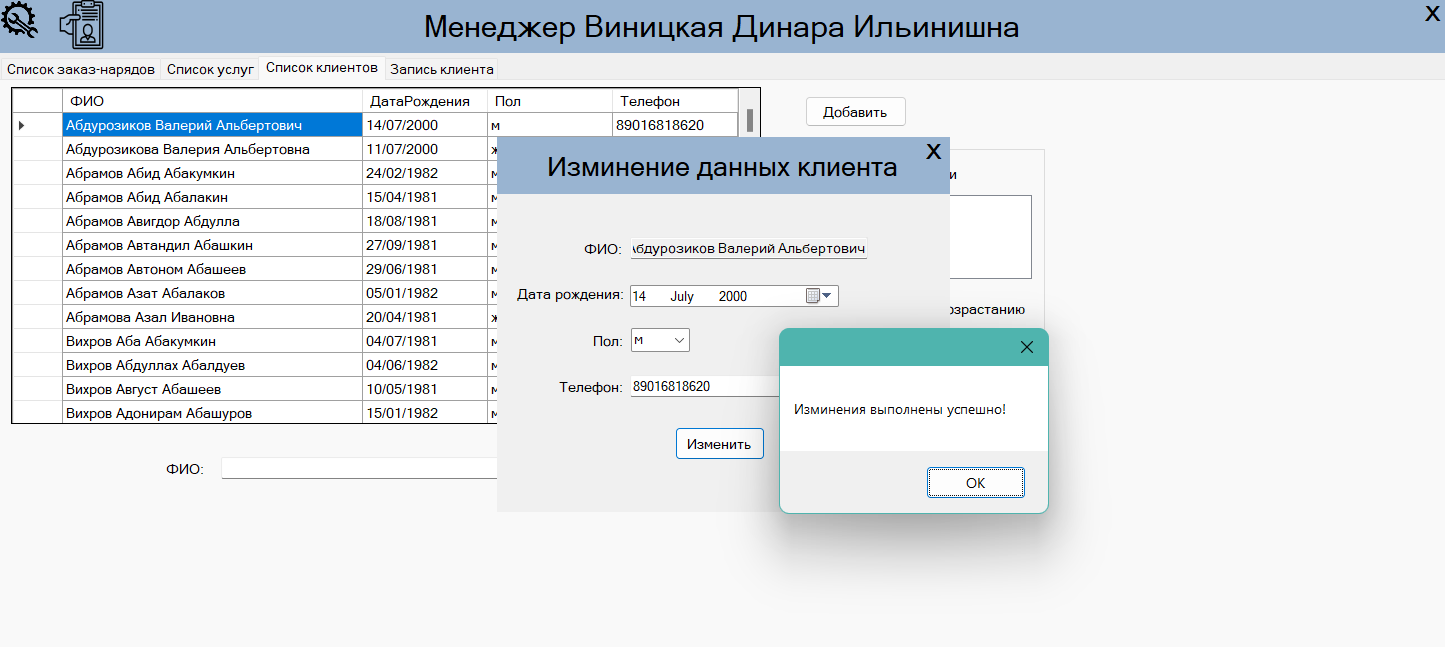


Рис. 2.30. Демонстрация корреткной работы формы “Manager\_edit\_client\_form”

В случае, если при введении данных была допущена ошибка, транзакция будет отклонена триггером “Check\_update” (см. Б.45) и пользователю выведется уведомление об ошибке (рис. 2.31).

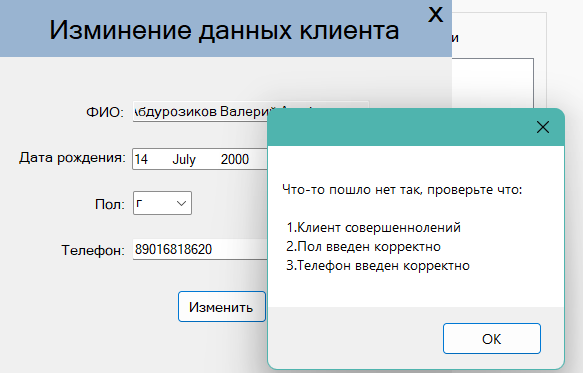


Рис. 2.31. Демонстрация работы триггера “Check\_update”

Если менеджер хочет добавить нового клиента в базу данных ему необходимо нажать на кнопку “Добавить” на соответствующие вкладке с клиентами, после чего будет вызвана форма “Manager\_add\_client\_form” (см. В.15). Демонстрация работы по добавлению нового клиента с помощью хранимой процедуры “Add\_new\_client” (см. Б.13) показана на скриншоте ниже (рис. 2.32).

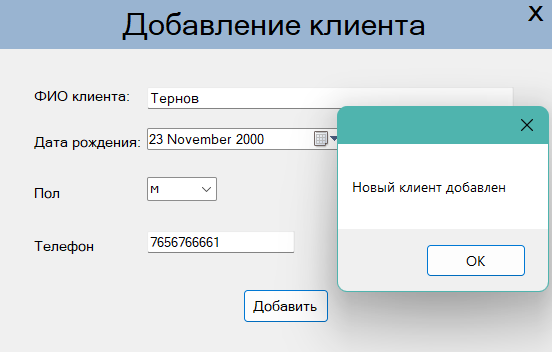


Рис. 2.32. Демонтарция корректной работы формы “Manager\_add\_client\_form”

В случае, если данные были введены некорректно транзакция будет отклонена одним из двух триггеров “Check\_client\_age” (см. Б. 47), “Check\_complex”(см. Б. 48). Срабатывание триггеров показано на скриншоте ниже (рис. 2.33).

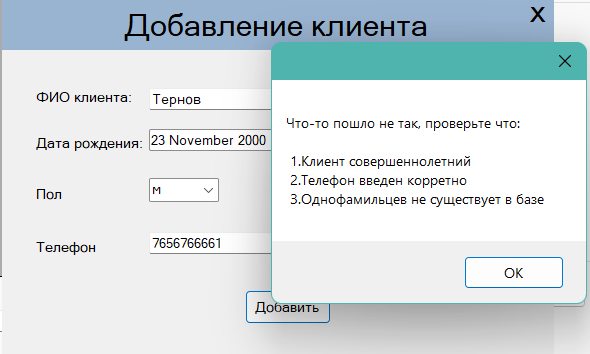


Рис. 2.33. Срабатывание триггеров “Check\_client\_age” и“Check\_complex”

На последней вкладке формы “Manager” запись клиента расположено две идентичные таблицы с данными о записях клиентов в автосервис. Разница в них заключается в том, что слева в таблице отображаются актуальные на сегодняшний записи, их можно отредактировать, либо удалить. Выводом данных о таких записях занимается хранимая процедура “Manager\_list\_of\_zapisi\_actual” (см. Б.14). Справа же находится список архивных записей минувших дней (рис. 2.34). Выводом информации об архивных записях занимается хранимая процедура “Manager\_list\_of\_zapisi” (см. Б.15). Также менеджеру доступна функция добавления новых записей.

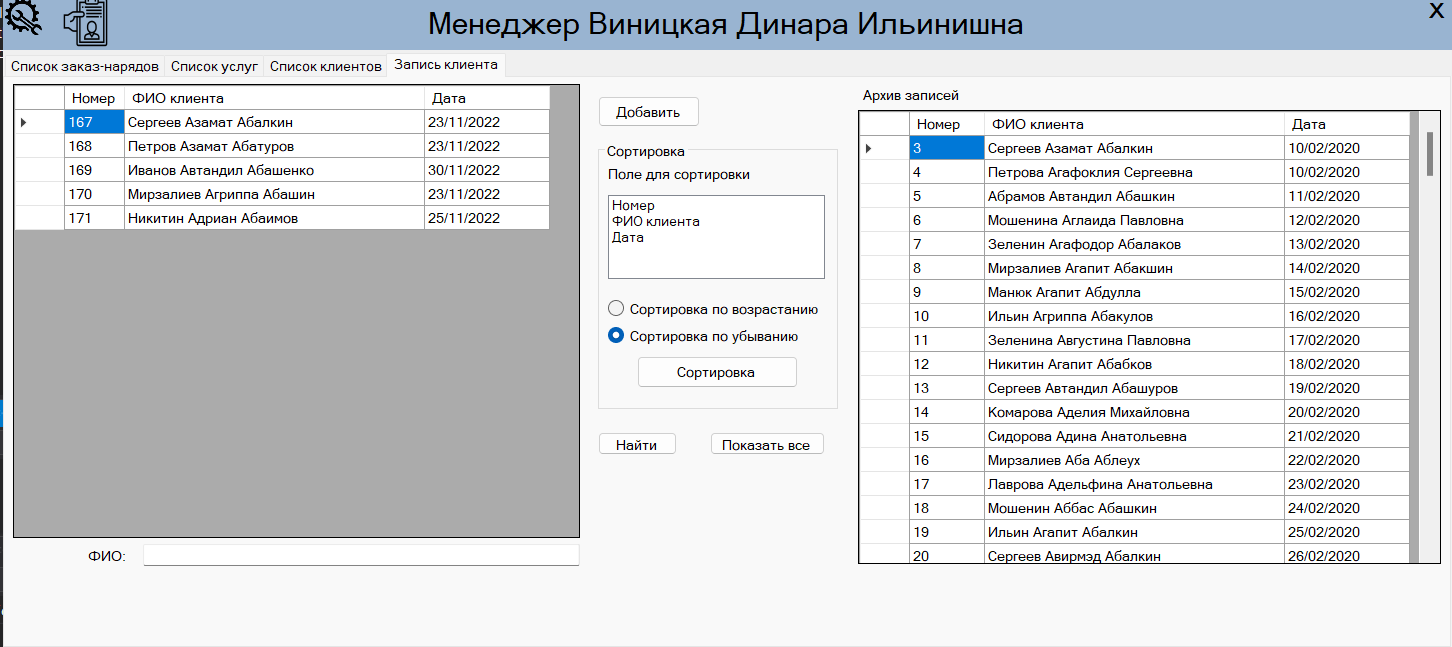


Рис. 2.34. Демонстрация работы формы “Manager” вкладка запись клиента

Менеджер имеет возможность создавать новые записи клиентов для этого ему необходимо нажать на кнопку “Добавить” расположенную между двумя таблицами, после чего будет вызвана соответствующая форма “Manager\_add\_zapisi\_form” (см. В.16) (рис. 2.35).

Добавление новой записи осуществляется с помощью хранимой процедуры “Manager\_add\_new\_zapisi” (см. Б.16).

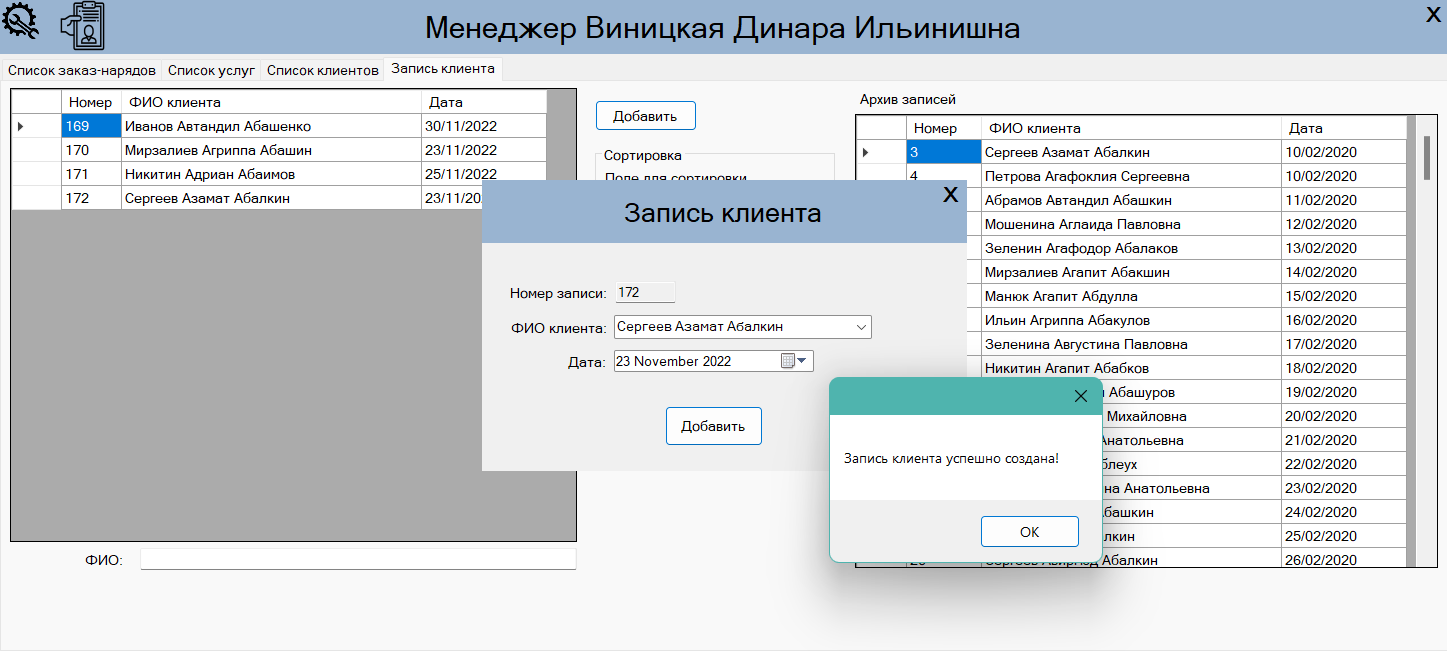


Рис. 2.35. Демонстрация работы формы “Manager\_add\_zapisi\_form”

Как было сказано выше менеджер имеет возможность отредактировать или удалить любую актуальную запись. Для этого ему необходимо дважды щелкнуть по ней и откроется соответствующая форма “Manager\_edit\_or\_delete\_zapisi\_form” (см. В.16) (рис 2. 36 – 2.37). Для выполнения вышеуказанных действий работают две хранимые процедуры, соответственно “Manager\_update\_zapasi” (см. Б.18) и “Manager\_delete\_zapisi” (см. Б.17).

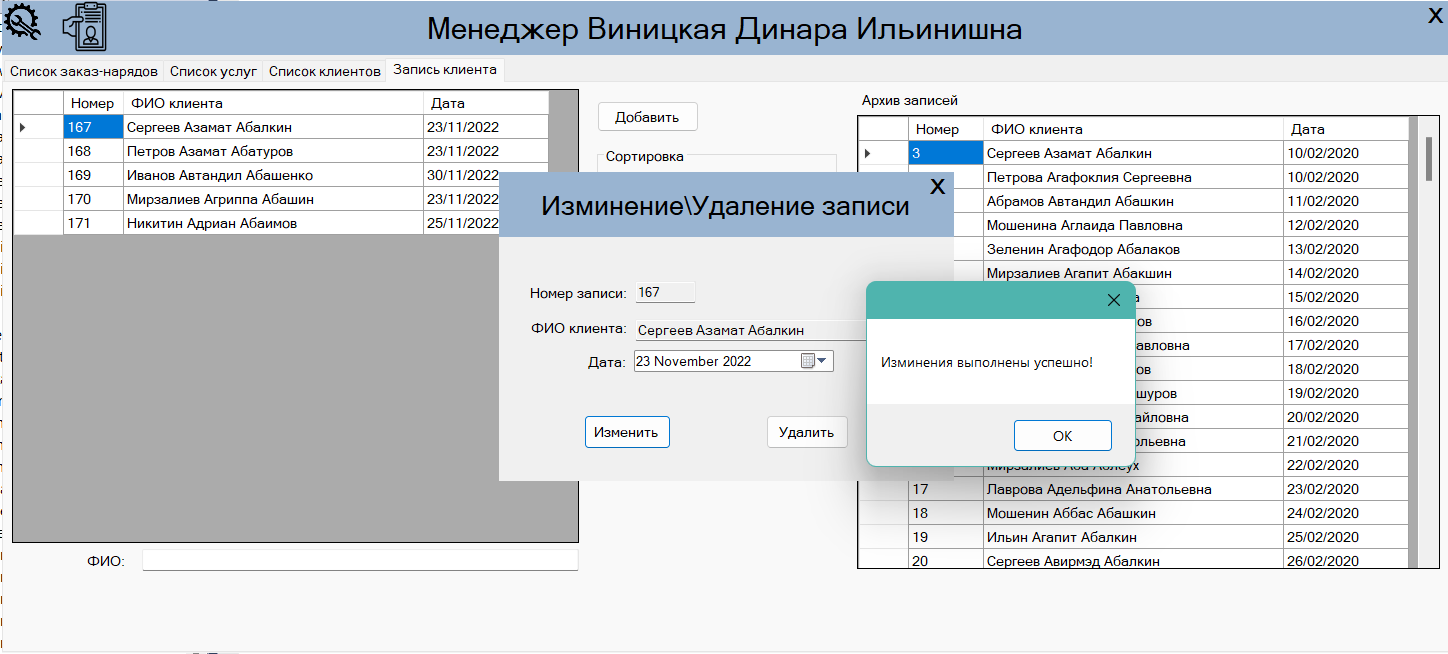


Рис. 2.36. Демонстрация работы формы “Manager\_edit\_or\_delete\_zapisi\_form” изменение

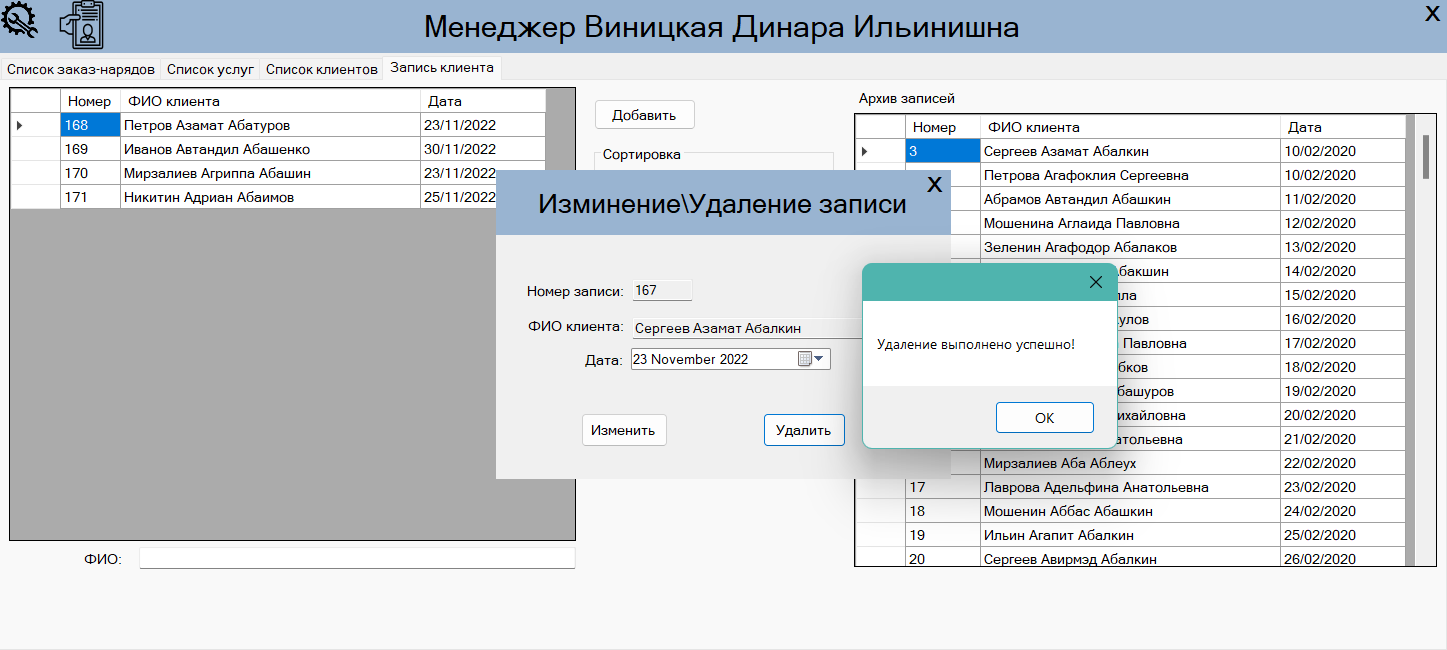


Рис. 2.37. Демонстрация работы формы “Manager\_edit\_or\_delete\_zapisi\_form” удаление

Также у менеджера, как и у всех других сотрудников есть доступ в личный кабинет, повторно рассматривать его не будем.

Если на этапе авторизации пользователь войдет в аккаунт директора, то откроется форма “Admin” (см. В.5) по структуре схожая с другими главными формами сотрудников. Здесь на первой вкладке “Работники” расположилась таблица с информацией о всех работающих в автосервисе людях. Выводом информации занимается хранимая процедура “admin\_list\_of\_workers” (см. Б.19) (рис. 2.38).



Рис. 2.38. Демонтрация работы формы “Admin” вкладка работники

Если директор автосервиса хочет добавить информацию о новом сотруднике в базу данных ему нужно нажать на кнопку “Добавить” после чего будет открыта новая форма “Admin\_add\_sort\_form” (см. Б.9). Введя все данные, если не было допущено ошибок хранимая функция “Admin\_add\_new\_sort” (см. Б.20) добавит работника в БД (рис. 2.39).

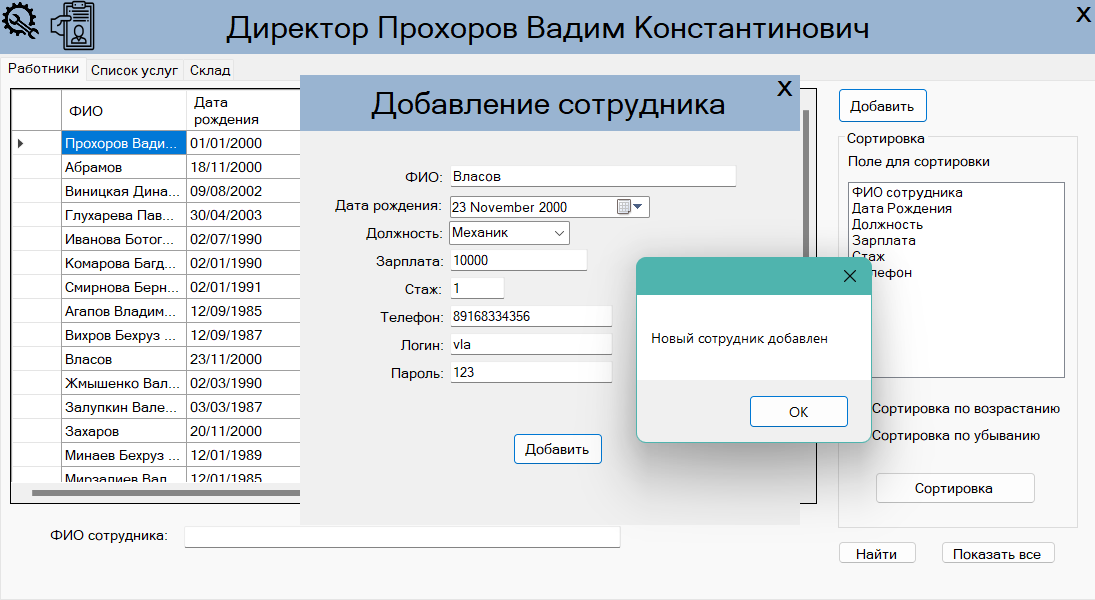


Рис. 2.39. Демонстрация корректной работы формы “Admin\_add\_sort\_form”

В случае, если при введении данных была допущена ошибка, транзакция будет отклонена триггером “Check\_sotr\_complex” (см. Б.49) (рис. 2.40).

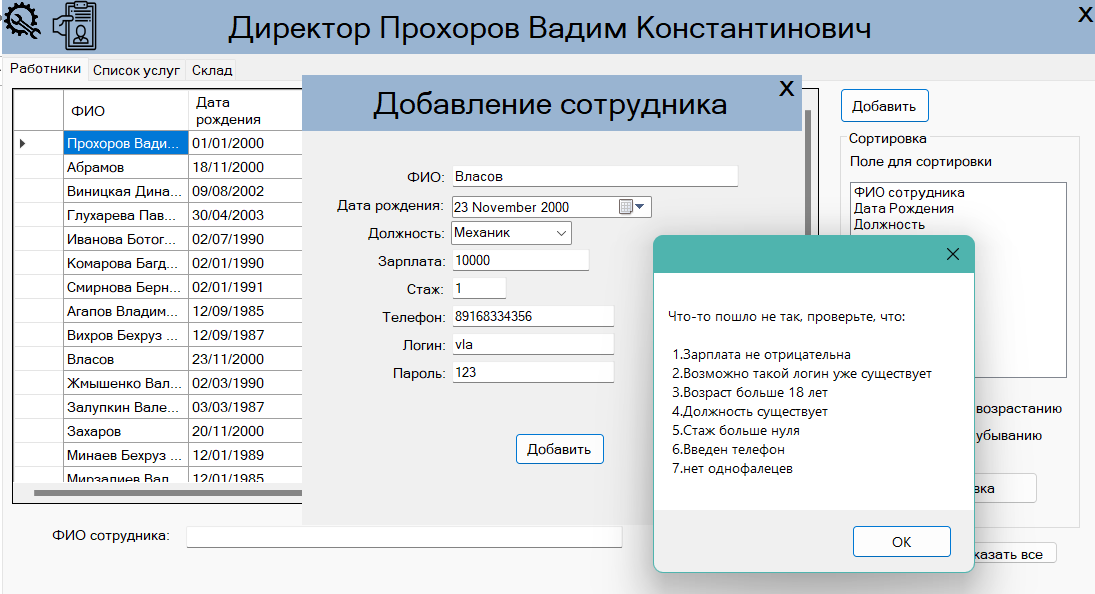


Рис. 2.41. Демонстрация работы триггера “Check\_sotr\_complex”

Если директор автосервиса хочет изменить информацию о сотруднике ему необходимо дважды щелкнуть по нему в списке сотрудников после чего откроется форма “Admin\_edit\_sotr\_form”. После изменения данных сотрудника и нажатия кнопки “Изменить” будет вызвана хранимая процедура “Admin\_edit\_sotr” (см. Б.21) которая с случае корректности изменений данных проведет данную транзакцию (рис. 2.42).

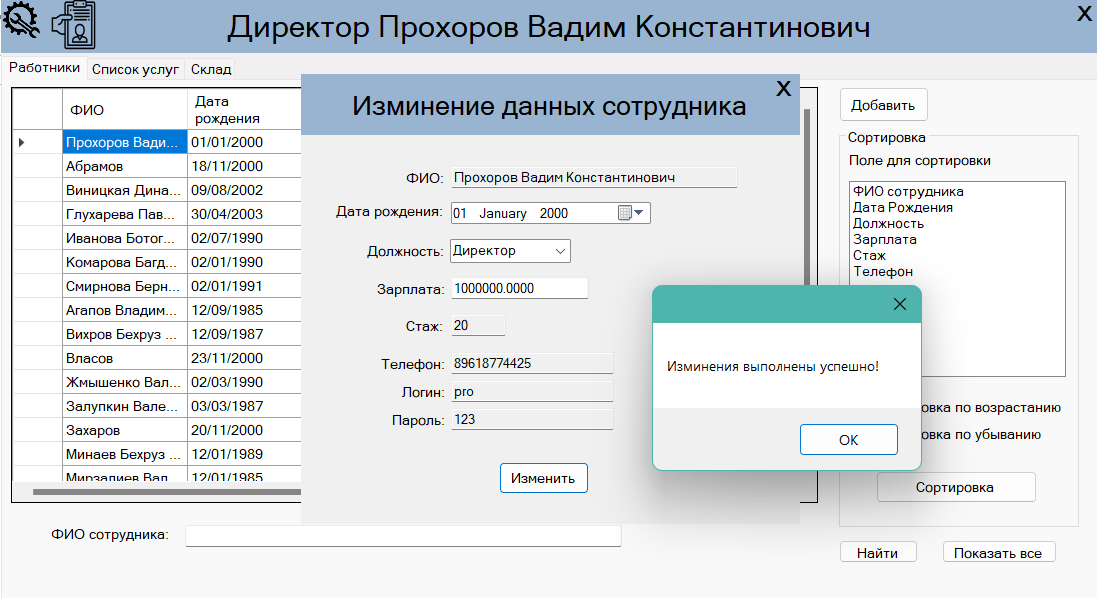


Рис. 2.42. Демонстрация корректной работы “Admin\_edit\_sotr\_form”

В случае, если при изменении данных были допущены ошибки транзакция будет прервана и пользователю будет выведено сообщение об ошибке благодаря триггеру “Check\_sotr\_update” (см. Б.44) (рис. 2.43).

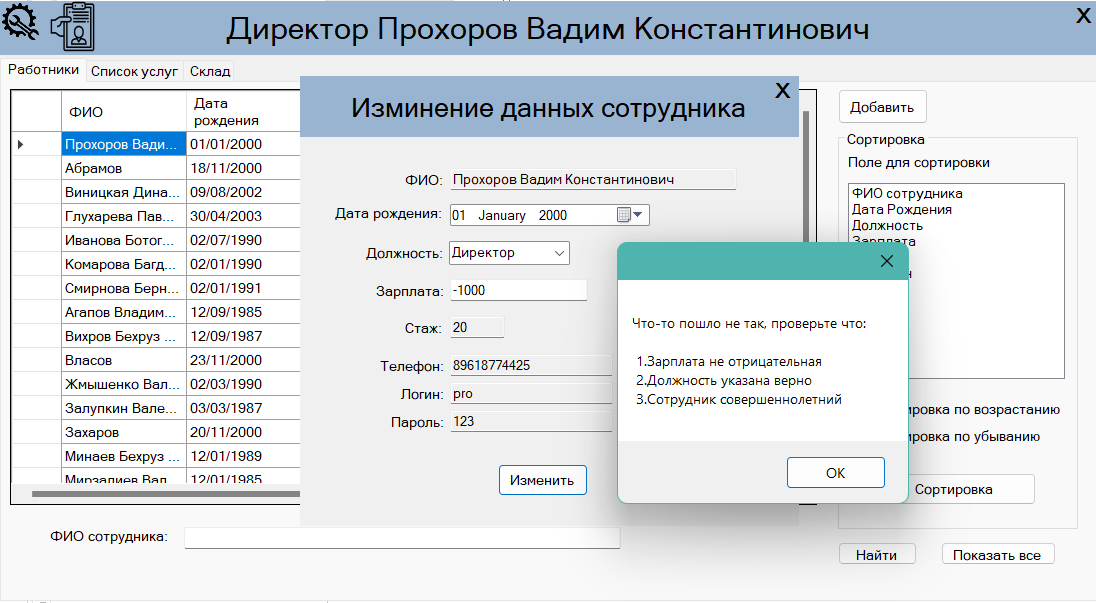


Рис. 2.43. Демонстрация работы триггера “Check\_sotr\_update”

На следующей вкладке “Список услуг” директор автосервиса увидит таблицу со всеми оказываемыми услугами и информацией о них. За вывод данных об услугах отвечает хранимая процедура “Admin\_list\_of\_service” (см. Б.10) (рис. 2.44).

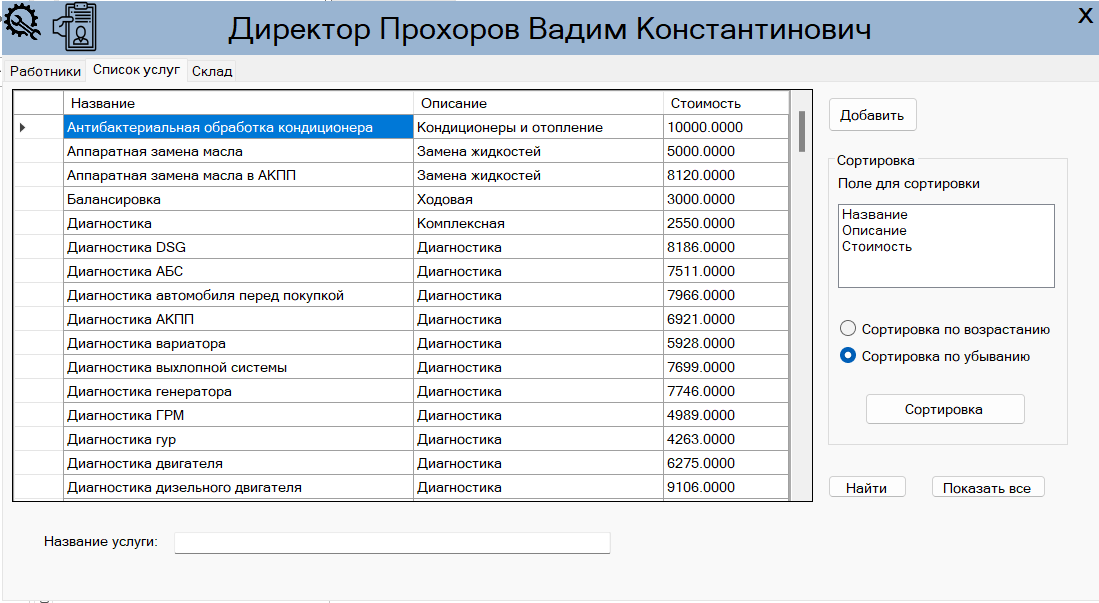


Рис. 2.44. Демонстрация работы вкладки “Список услуг”

Если директор автосервиса захочет добавить новую услугу ему необходимо нажать на кпоку “Добавить” после чего будет вызвана форма “Admin\_add\_service\_form” (см. В.8). После введения данных о новой услуге и нажатия кнопки “Добавить” будет вызвана хранимая процедура “Admin\_add\_service” (см. Б.22) (рис. 2.45).

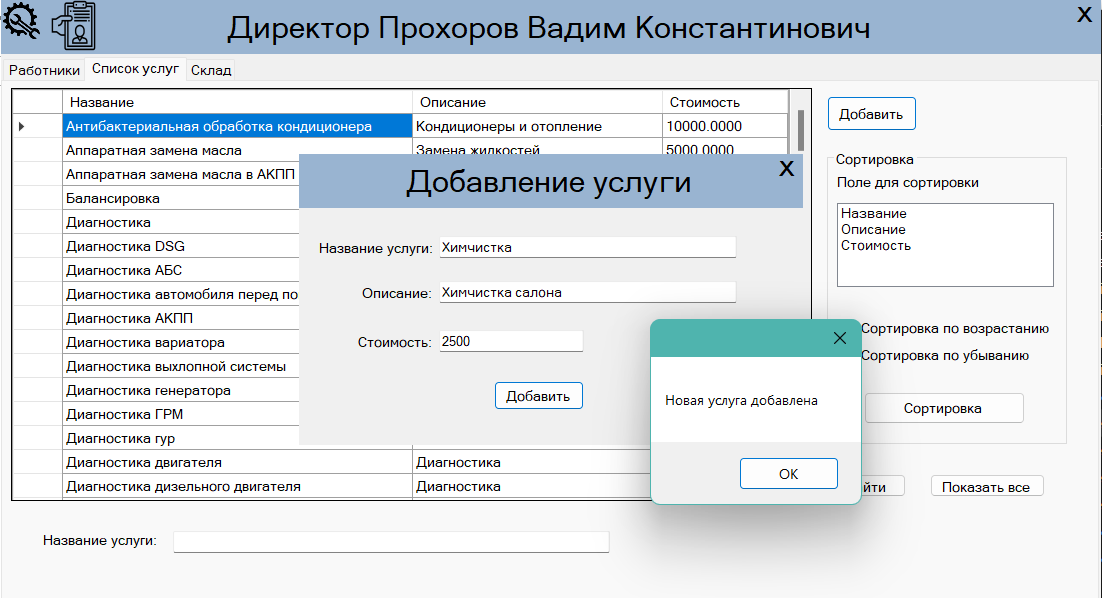


Рис. 2.45. Демонстрация работы формы “Admin\_add\_service\_form”

В случае, если при добавлении новой услуги во вресенных данных была допущена ошибка транзацкция будет отменена триггером “Check\_cost\_service” (см. Б.50) (рис. 2.46).

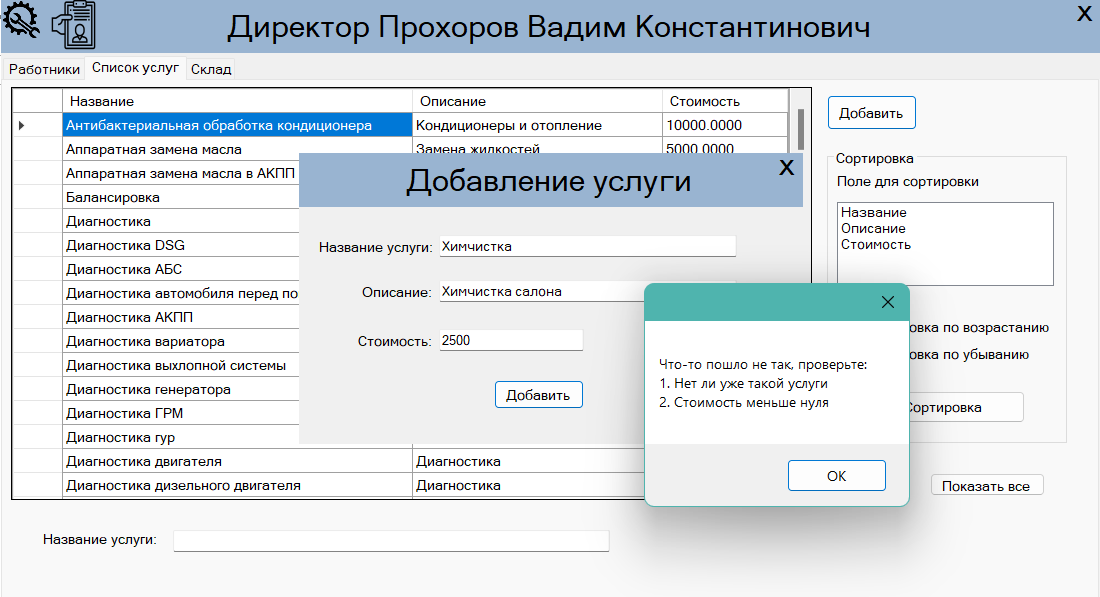


Рис. 2.46. Демонстрация работы триггера “Check\_cost\_service”

Если директор хочет изменить стоимость услуги ему необходимо дважды кликнуть по соответствующей услуге после чего откроется форма “Admin\_edit\_service\_form” (см. Б.11). За применение вносимых изменений отвечает хранимая процедура “Admin\_edit\_service” (рис. 2.47).

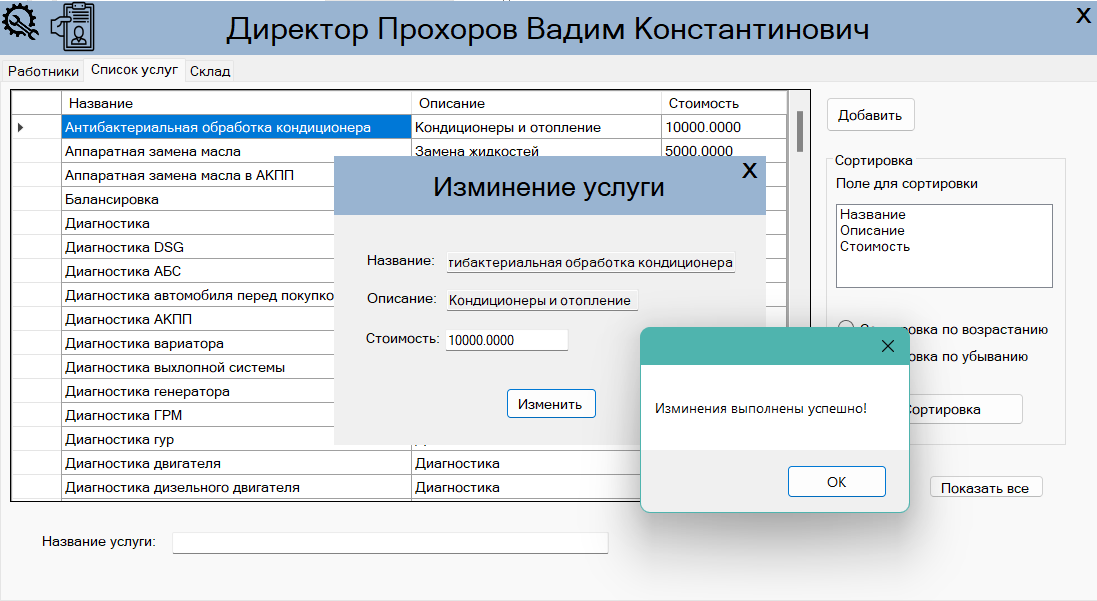


Рис. 2.48. Демонстрация корреткной работы формы“Admin\_edit\_service\_form”

В случае, если стоимость была введена некорректно транзакция будет отклонена триггером “Check\_cost\_update\_service” (см. Б.51), а пользователь увидит сообщение об ошибке (рис. 2.49).

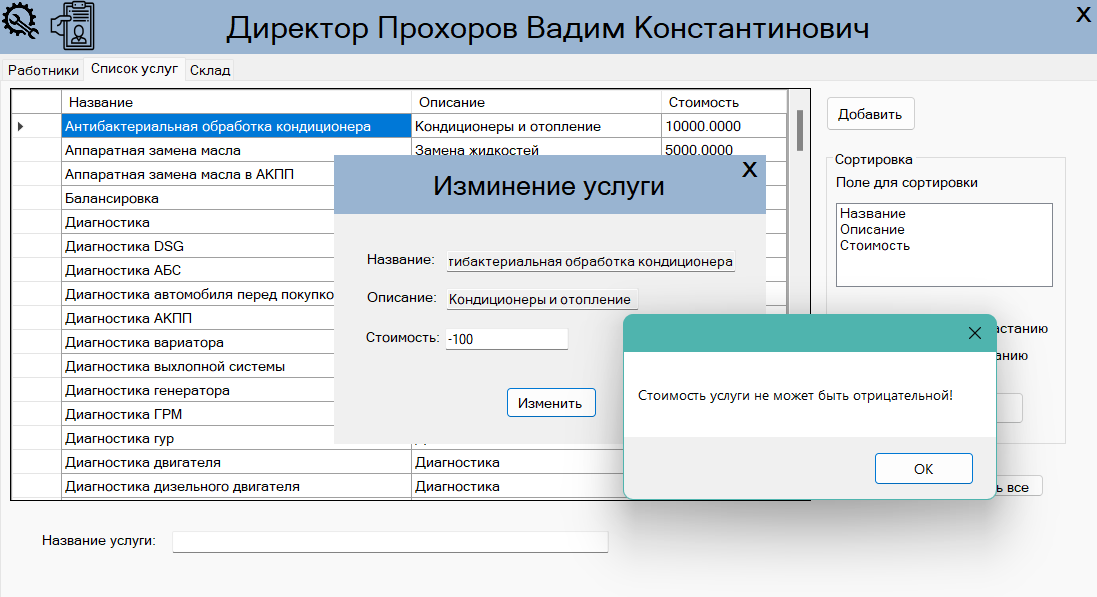


Рис. 2.49. Демонстрация работы триггера “Check\_cost\_update\_service”

Если директор щелкнет по вкладке “Склад”, то увидит перечень хранимых запчастей и материалов на складе. За вывод данных в соответствующую таблицу отвечает хранимая процедура “Admin\_list\_of\_detali” (см. Б.24) (рис. 2.50).

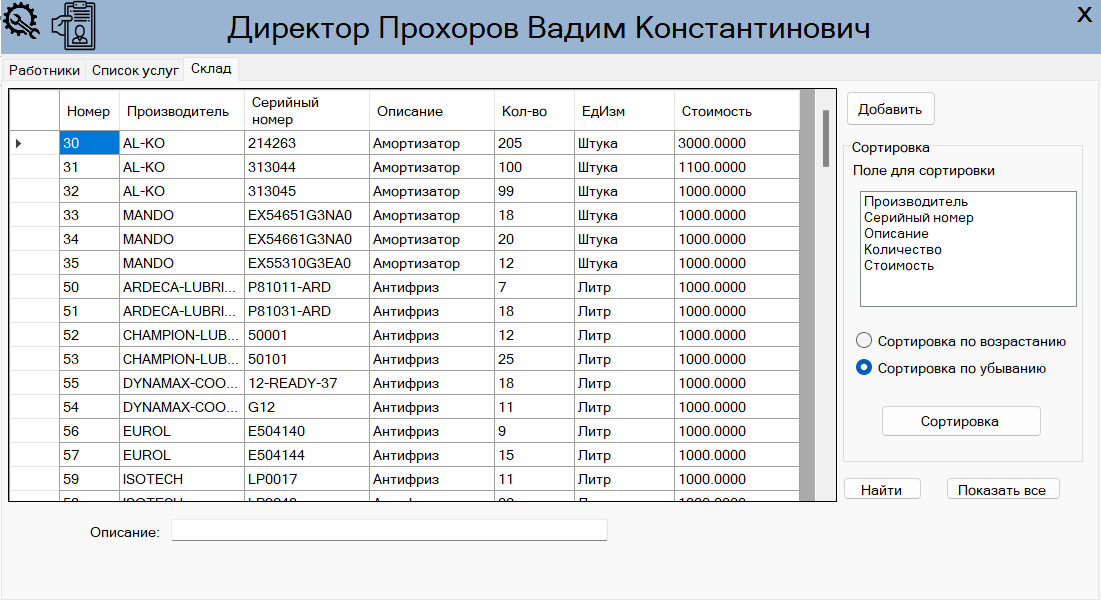


Рис. 2.50. Демонстрация работы вкладки склад

Если директор автосервиса захочет добавить новую деталь ему необходимо нажать на кнопку “Добавить” после чего будет вызвана форма “Admin\_add\_detali\_form”. После введения данных о новой услуге и нажатия кнопки “Добавить” будет вызвана хранимая процедура “Admin\_add\_detali” (рис. 2.51).

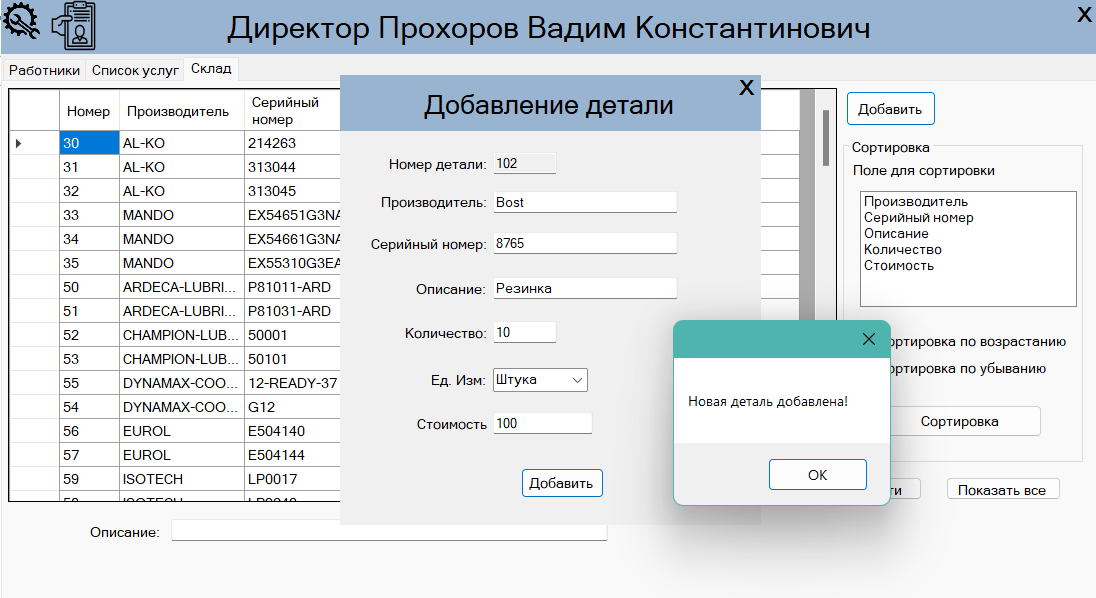


Рис. 2.51. Демонстрация работы формы “Admin\_add\_detali\_form”

В случае, если при добавлении новой детали во внесённых данных была допущена ошибка транзакция будет отменена триггером “Check\_cost\_count\_detali” (см. Б.54) (рис. 2.52).

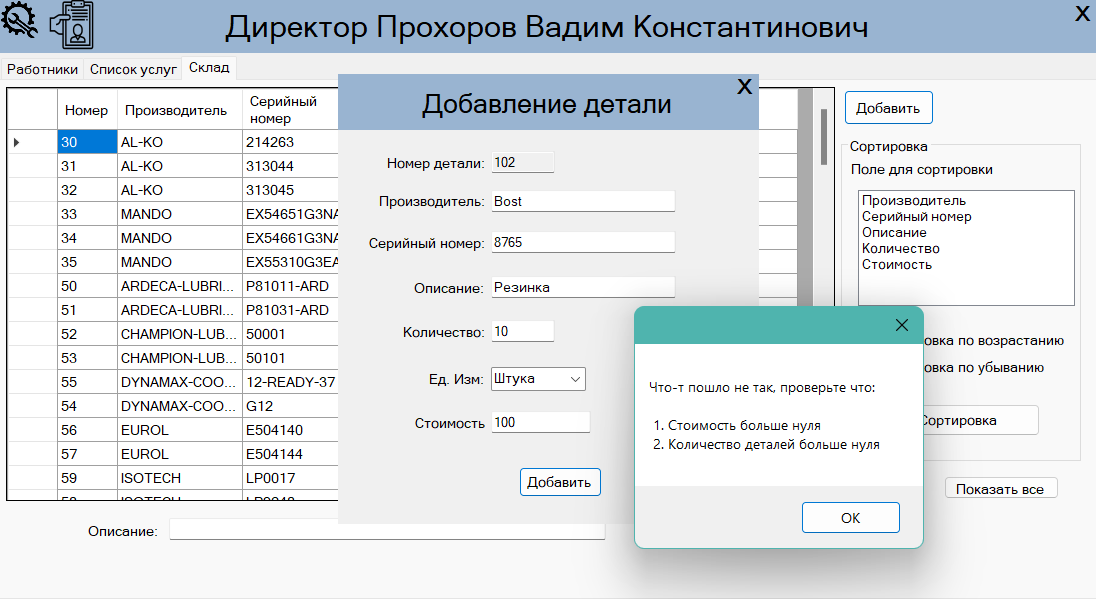


Рис. 2.52. Демонстрация работы триггера “Check\_cost\_count\_detali”

Если директор хочет изменить стоимость уже имеющейся в списке детали ему необходимо дважды кликнуть по соответствующей дели после чего откроется форма “Admin\_edit\_detali\_form” (см. В.10). За применение вносимых изменений отвечает хранимая процедура “Admin\_edit\_detali” (рис. 2.53).

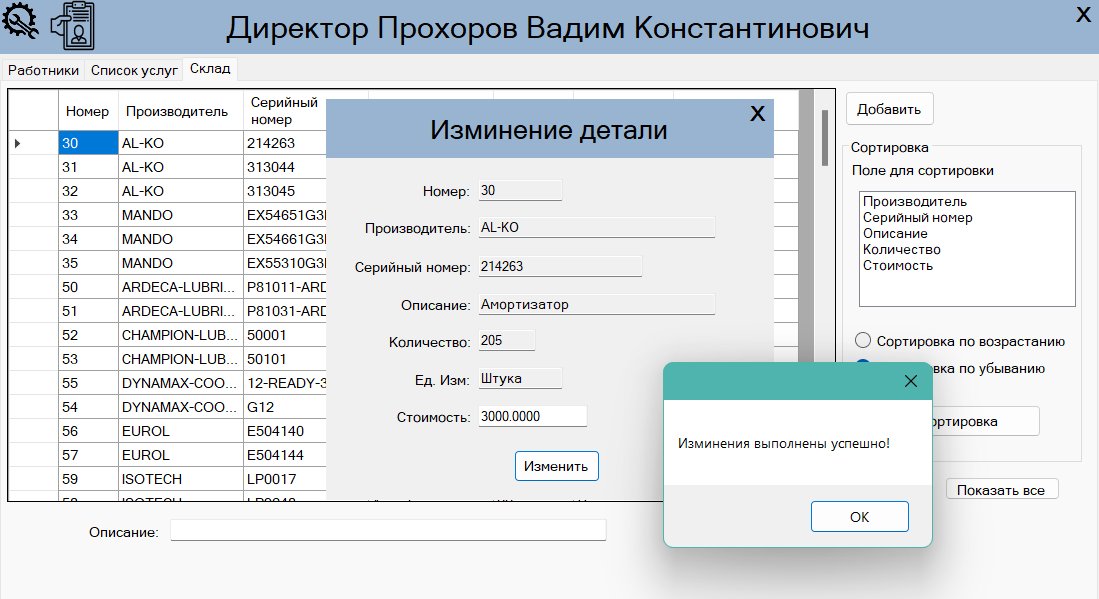


Рис. 2.53. Демонстрация корреткной работы формы“Admin\_edit\_detali\_form”

В случае, если стоимость была введена некорректно транзакция будет отклонена одним из триггеров “Check\_count\_detali” (см. Б.43) или “Check\_cost\_update” (см. Б.51), а пользователь увидит сообщение об ошибке (рис. 2.54).

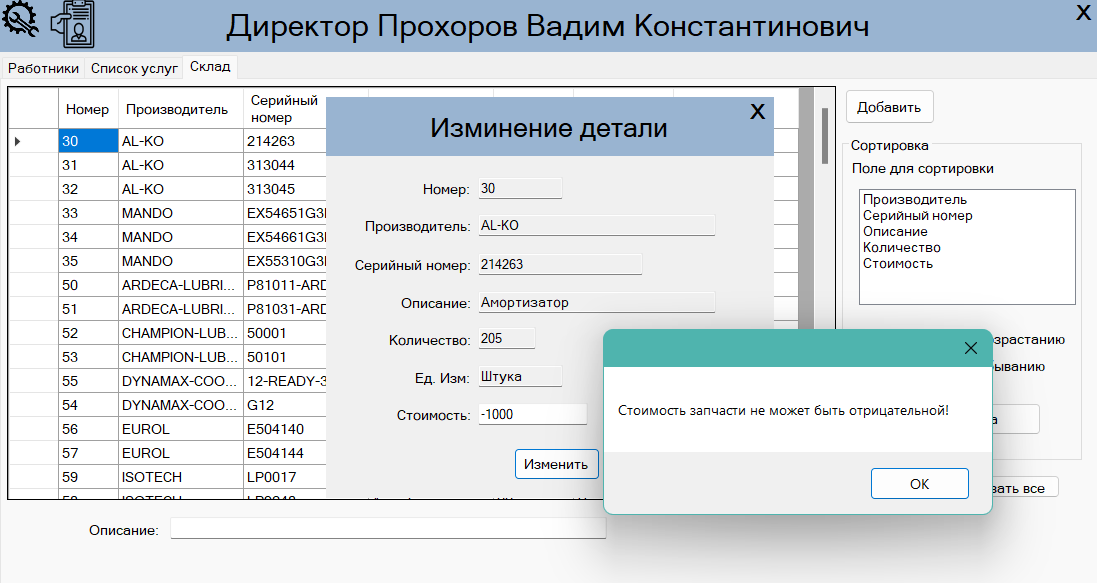


Рис. 2.54. Демонстрация работы триггеров “Check\_count\_detali” и “Check\_cost\_update”

Заключение

Для курсовой работы была выбрана предметная область «Автосервис» с целью проектирования базы данных, которая оптимизирует работу автосервиса.

Деятельность по автоматизации многих функций в автосервисе нельзя переоценить, а создаваемая база данных станет замечательным и, главное, актуальным инструментом, который в руках специалистов позволит ускорить и вывести качество предоставляемых услуг на новый уровень.

В ходе выполнения самостоятельной работы были получены следующие результаты:

1. При анализе предметной области были рассмотрены программные продукты, решающие задачи в этой области (“ПростойСофт Автосервис” и “1С:Предприятие 8. Автосервис”), изучены функции, реализованные в данных продуктах. Итогом анализа стал список функций, которые планируется реализовать в данной работе.
2. На этапе концептуального проектирования были выделены основные пользователи системы (Механик, Менеджер, Директор), определены их функции, а также построена и описана диаграммы вариантов использования. Итогом проделанной работы стали выделенные объекты, информация о которых будет храниться в разрабатываемой базе данных.
3. На этапе логического проектирования была разработана логическая модель базы данных автосервиса, включающая в себя ER-диаграммы. По правилам анализа ER-диаграмм сформированы основные отношения (Заказ-наряд, Транспортное средство, Деталь, Клиент, Услуга, Запись, День, Работник, Аналог, Расход, УслугиЗН, Смена ), хранящие информацию о деятельности автосервиса, в которые добавлены неключевые атрибуты.
4. На этапе физического проектирования была рассмотрена среда SQL Server Manager, позволяющая работать с СУБД MS SQL Server в более удобном графическом интерфейсе. Вместе с этим были определены требования к таблицам и схема данных, отражающая связи между таблицами. В итоге были созданы структуры всех объектов проектируемой базы данных, а таблицы были заполнены тестовыми данными. Созданы хранимые функции, реализующий теоретически заложенный функционал программы.
5. На этапе описания перечня функций были перечислены все возможности реализованные в рамках написания программы для данного курсового проекта.
6. На этапе описания работы была проведена масштабная работа по составлению сценариев использования реализованного программного средства. Были описаны все имеющиеся формы, способы взаимодействия с функциями, сценарии при которых пользователь увидит сообщения об ошибках и варианты их исправления.

Итогом курсового проекта стало разработанное программное средство, программа, автоматизирующая процессы, протекающие в автосервисе. Программа была протестирована на многих примерах и показала отличный результат отказоустойчивости. Это приложение можно использовать на практике и в будущем можно сделать его доступным для массового использования. В перспективе эту программу также можно дополнить различными методами и функциями.

Список используемых источников

1. Файловый архив студентов StudFiles [Электронный ресурс] – Режим доступа: https://studfile.net/preview/9553006/page:6/, свободный. Дата обращения: 25.11.2022 г.
2. Национальный открытый университет ИНТУИТ [Электронный ресурс] – Режим доступа https://intuit.ru/studies/courses/574/430/lecture/9749, свободный. Дата обращения: 25.11.2022 г.
3. Сайт “1C:Предприятие 8. Автосервис” [Электронный ресурс] – Режим доступа: https://solutions.1c.ru/catalog/autoservice/materials, свободный. Дата обращения: 25.11.2022 г.
4. Сайт “Простой Софт” [Электронный ресурс] – Режим доступа: https://prostoysoft.ru/AutoService.htm, свободный. Дата обращения: 25.11.2022 г.
5. Файловый архив Bstudy [Электронный ресурс] – Режим доступа: https://bstudy.net/733745/informatika/proektirovanie\_bazy\_dannyh, свободный. Дата обращения: 25.11.2022 г.
6. Файловый архив Studbooks [Электронный ресурс] – Режим доступа: https://studbooks.net/2081093/informatika/logicheskoe\_proektirovanie\_bazy\_dannyh, свободный. Дата обращения: 25.11.2022 г.

Приложение А. Примеры заполнения таблиц

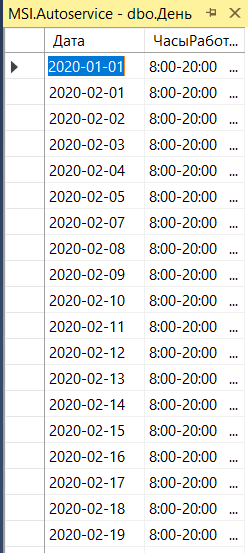


Рис. A.1. Данные таблицы «День»

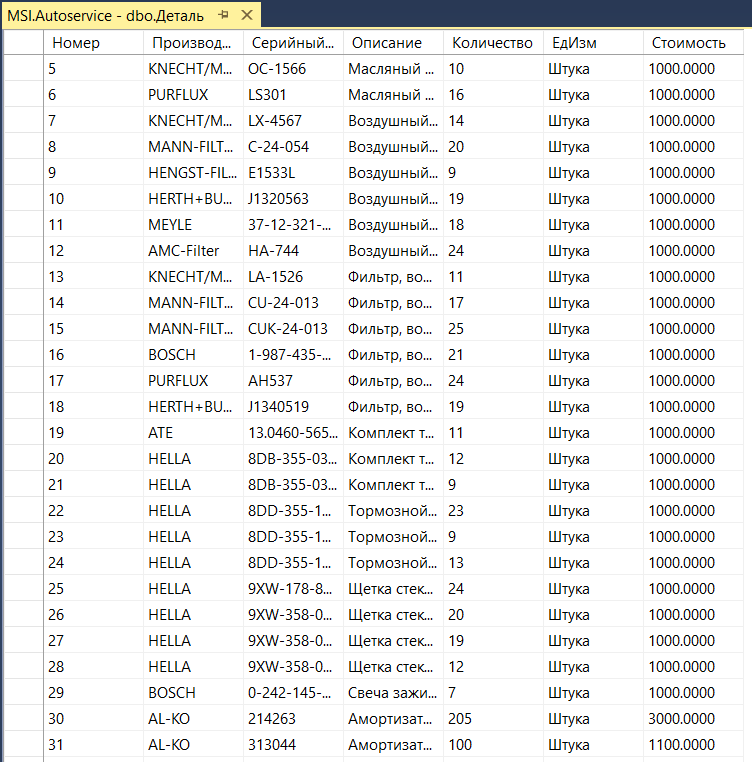


Рис. A.2. Данные таблицы «Деталь»

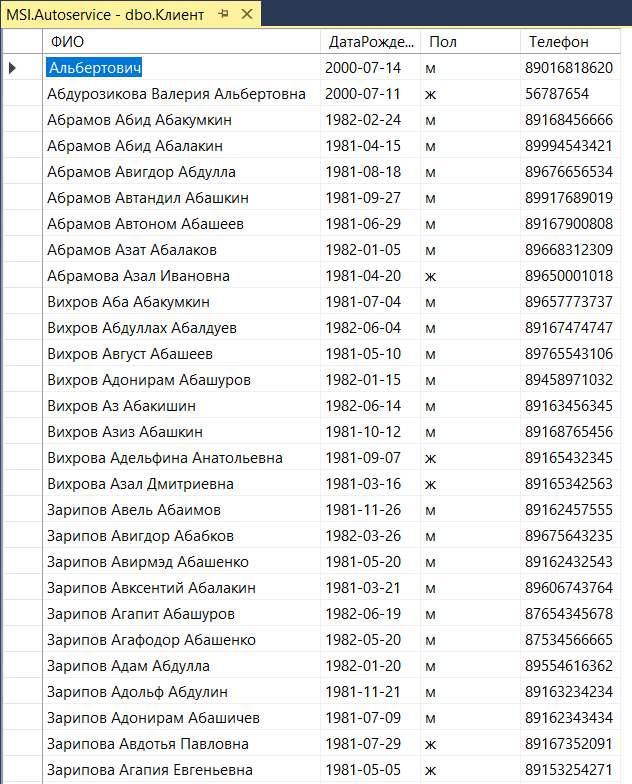


Рис. А.3. Данные таблицы «Клиент»

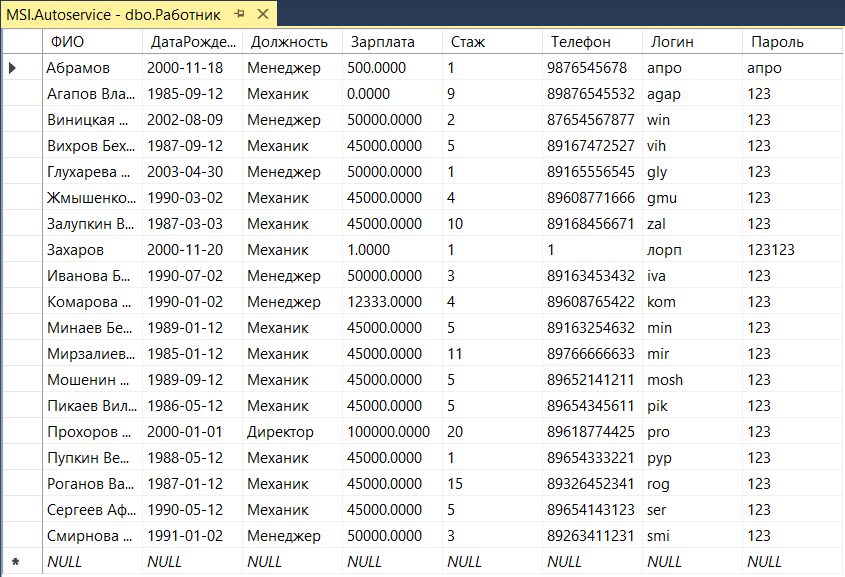


Рис. А.4. Данные таблицы «Работник»

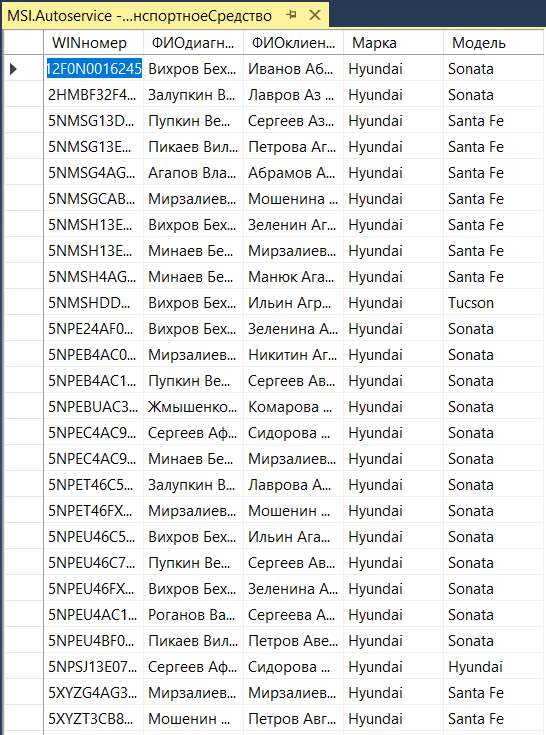


Рис. А.5. Данные таблицы «ТранспортноеСредство»

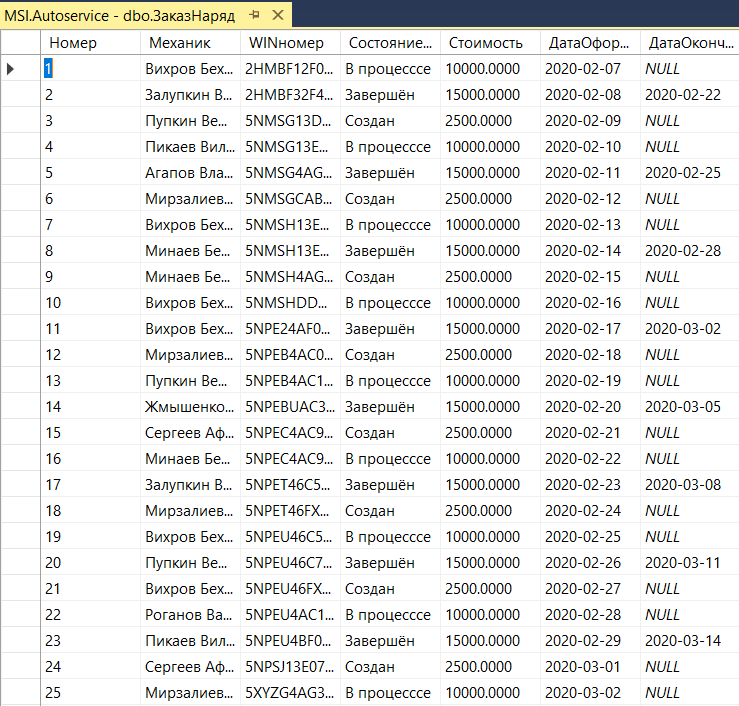


Рис. А.6. Данные таблицы «ЗаказНаряд»

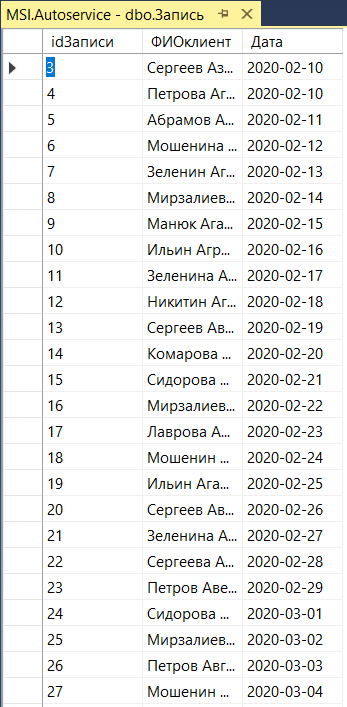


Рис. А.7. Данные таблицы «Запись»

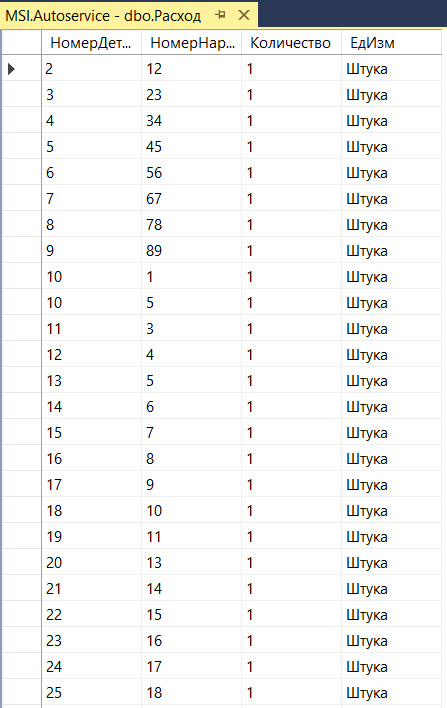


Рис. А.8. Данные таблицы «Расход»

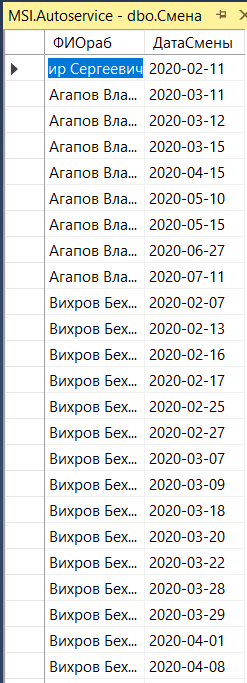


Рис. А.9. Данные таблицы «Смена»



Рис. А.10. Данные таблицы «Услуга»

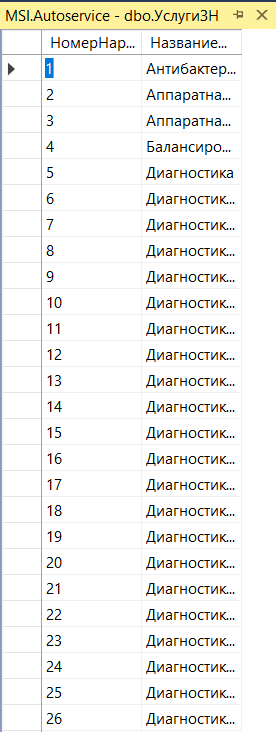


Рис. А.11. Данные таблицы «УслугиЗН»

Приложение Б. SQL-операторы создания объектов БД

Хранимые процедуры

account\_data Б.1

ALTER PROCEDURE [dbo].[account\_data]

@name varchar(50)

AS

BEGIN

select ДатаРождения,Должность,Зарплата,Стаж,Телефон,Логин,Пароль from Работник

where ФИО = @name

END

account\_edit\_data Б.2

Create PROCEDURE [dbo].[account\_edit\_data]

@fio varchar(50), @date date, @teleph bigint, @login varchar(10), @pass varchar(10)

AS

BEGIN

update Работник set ДатаРождения = @date, Телефон = @teleph, Логин = @login, Пароль = @pass where ФИО = @fio

END

last\_num Б.3

ALTER PROCEDURE [dbo].[last\_num]

-- Add the parameters for the stored procedure here

@num int output

AS

BEGIN

select @num = (max(Номер)+1) from ЗаказНаряд

END

diagnostic\_cost Б.4

ALTER PROCEDURE [dbo].[diagnostic\_cost]

@cost int output

AS

BEGIN

select @cost = Стоимость from Услуга where Название = 'Диагностика'

END

list\_of\_detali Б.5

ALTER PROCEDURE [dbo].[list\_of\_detali]

@number int

AS

BEGIN

select Деталь.Описание as [Затраченная деталь], Деталь.Производитель,Деталь.[Серийный номер] ,Расход.Количество, Деталь.Стоимость from Расход

join ЗаказНаряд on ЗаказНаряд.Номер = Расход.НомерНаряда

join Деталь on Расход.НомерДетали = Деталь.Номер

where ЗаказНаряд.Номер = @number

END

list\_of\_service Б.6

ALTER PROCEDURE [dbo].[list\_of\_service]

@number int

AS

BEGIN

select УслугиЗН.НазваниеУслуги as [Название услуги], Услуга.Стоимость from УслугиЗН

join ЗаказНаряд on УслугиЗН.НомерНаряда = ЗаказНаряд.Номер

join Услуга on Услуга.Название = УслугиЗН.НазваниеУслуги

where ЗаказНаряд.Номер = @number

END

add\_service\_to\_zakaz Б.7

ALTER PROCEDURE [dbo].[add\_service\_to\_zakaz]

@num\_zakaz int, @name\_service varchar(50)

AS

BEGIN

insert into УслугиЗН values(@num\_zakaz,@name\_service)

END

Mechanic\_new\_car Б.8

ALTER PROCEDURE [dbo].[Mechanic\_new\_car]

@win varchar(50), @fioMech varchar(50), @fioClient varchar(50), @fab varchar(50), @model varchar(50)

AS

BEGIN

insert into ТранспортноеСредство values(@win, @fioMech , @fioClient , @fab, @model)

END

Manager\_list\_of\_zakaz Б.9

ALTER PROCEDURE [dbo].[Manager\_list\_of\_zakaz]

AS

BEGIN

select

ЗаказНаряд.Номер, ЗаказНаряд.Механик, ЗаказНаряд.WINномер,

ЗаказНаряд.СостояниеЗаказа,ЗаказНаряд.ДатаОформления,

ЗаказНаряд.ДатаОкончания,ЗаказНаряд.Стоимость, ТранспортноеСредство.ФИОклиента, ТранспортноеСредство.Марка, ТранспортноеСредство.Модель

from ЗаказНаряд

join ТранспортноеСредство on ЗаказНаряд.WINномер = ТранспортноеСредство.WINномер

order by ЗаказНаряд.СостояниеЗаказа

end

Admin\_list\_of\_service Б.10

ALTER PROCEDURE [dbo].[Admin\_list\_of\_service]

AS

BEGIN

select \* from Услуга order by Название

END

Manager\_list\_of\_clients Б.11

ALTER PROCEDURE [dbo].[Manager\_list\_of\_clients]

AS

BEGIN

select \* from Клиент order by ФИО

END

manager\_edit\_client Б.12

ALTER PROCEDURE [dbo].[manager\_edit\_client]

@fio varchar(50),@date date,@gender char(1),@teleph bigint

AS

BEGIN

update Клиент set ДатаРождения = @date, Пол = @gender, Телефон = @teleph

where ФИО = @fio

END

Add\_new\_client Б.13

ALTER PROCEDURE [dbo].[Add\_new\_client]

@fio varchar(50), @date date, @gender char(1), @tel bigint

AS

BEGIN

insert into Клиент Values(@fio, @date, @gender, @tel)

END

Manager\_list\_of\_zapisi\_actual Б.14

ALTER PROCEDURE [dbo].[Manager\_list\_of\_zapisi\_actual]

AS

BEGIN

select \* from Запись where Дата < getdate()

END

Manager\_list\_of\_zapisi Б.15

ALTER PROCEDURE [dbo].[Manager\_list\_of\_zapisi]

AS

BEGIN

select \* from Запись where DATEADD(day, 1, Дата) >= getdate()

END

Manager\_add\_new\_zapisi Б.16

ALTER PROCEDURE [dbo].[Manager\_add\_new\_zapisi]

@num int, @fio varchar(50), @date date

AS

BEGIN

insert into Запись values(@num,@fio,@date)

END

Manager\_delete\_zapisi Б.17

ALTER PROCEDURE [dbo].[Manager\_delete\_zapisi]

@num int

AS

BEGIN

delete from Запись where idЗаписи = @num

END

Manager\_update\_zapasi Б.18

ALTER PROCEDURE [dbo].[Manager\_update\_zapasi]

@num int, @date date

AS

BEGIN

update Запись set Дата = @date where idЗаписи = @num

END

Admin\_list\_of\_workers Б.19

ALTER PROCEDURE [dbo].[Admin\_list\_of\_workers]

AS

BEGIN

select \* from Работник order by Должность, ФИО

END

Admin\_add\_new\_sort Б.20

ALTER PROCEDURE [dbo].[Admin\_add\_new\_sort]

@fio varchar(50), @date date, @state varchar(50), @salary money, @stage int, @teleph bigint, @login varchar(10), @pass varchar(10)

AS

BEGIN

insert into Работник values(@fio, @date, @state, @salary, @stage, @teleph, @login, @pass)

END

Admin\_edit\_sotr Б.21

ALTER PROCEDURE [dbo].[Admin\_edit\_sotr]

@fio varchar(50), @pos varchar(50), @salary money

AS

BEGIN

update Работник set Должность = @pos, Зарплата = @salary where ФИО = @fio

END

Admin\_add\_service Б.22

ALTER PROCEDURE [dbo].[Admin\_add\_service]

@name varchar(50), @description varchar(50), @cost money

AS

BEGIN

insert into Услуга values(@name,@description,@cost)

END

Admin\_edit\_service Б.23

ALTER PROCEDURE [dbo].[Admin\_edit\_service]

@name varchar(50), @des varchar(50), @cost money

AS

BEGIN

update Услуга set Стоимость = @cost where Название = @name and Описание = @des

END

Admin\_list\_of\_detali Б.24

ALTER PROCEDURE [dbo].[Admin\_list\_of\_detali]

AS

BEGIN

select \* from Деталь order by Описание, Производитель, [Серийный номер]

END

Add\_detali\_to\_zakaz Б.25

ALTER PROCEDURE [dbo].[Add\_detali\_to\_zakaz]

@num\_zakaz int, @count int, @format varchar(5),@discription varchar(50), @fab varchar(50), @series varchar(50)

AS

BEGIN

declare @num\_det int

select @num\_det = Номер from Деталь where Описание = @discription and Производитель = @fab and [Серийный номер] = @series

insert into Расход values (@num\_det, @num\_zakaz, @count, @format)

update Деталь set Количество = Количество - @count where Номер = @num\_det

END

add\_new\_zakaz Б.26

ALTER PROCEDURE [dbo].[add\_new\_zakaz]

@num int, @name varchar(50), @win varchar(50), @sost varchar(50), @cost money,@date date

AS

BEGIN

insert into ЗаказНаряд (Номер, Механик, WINномер, СостояниеЗаказа,Стоимость,ДатаОформления) values (@num, @name, @win, @sost, @cost, @date)

insert into УслугиЗН values (@num,'Диагностика')

END

admin\_add\_detali Б.27

ALTER PROCEDURE [dbo].[admin\_add\_detali]

@num int, @fab varchar(50), @series varchar(50),@description varchar(50), @count int, @EdIzm varchar(5), @cost money

AS

BEGIN

insert into Деталь values (@num, @fab, @series, @description, @count, @EdIzm, @cost)

END

Admin\_add\_service Б.28

ALTER PROCEDURE [dbo].[Admin\_add\_service]

@name varchar(50), @description varchar(50), @cost money

AS

BEGIN

insert into Услуга values(@name,@description,@cost)

END

Admin\_list\_of\_workers Б.29

ALTER PROCEDURE [dbo].[Admin\_list\_of\_workers]

AS

BEGIN

select \* from Работник order by Должность, ФИО

END

Edit\_zakaz Б.30

ALTER PROCEDURE [dbo].[Edit\_zakaz]

@name varchar(50), @sost varchar(50), @cost money,@date date, @num int

AS

BEGIN

update ЗаказНаряд set Механик=@name,СостояниеЗаказа=@sost,Стоимость=@cost,ДатаОкончания=@date

where Номер = @num

END

filter\_wincode Б.31

ALTER PROCEDURE [dbo].[filter\_wincode]

@win varchar(10)

AS

BEGIN

select WINномер from ТранспортноеСредство where WINномер like @win + '%'

END

last\_num\_detali Б.32

ALTER PROCEDURE [dbo].[last\_num\_detali]

@num int output

AS

BEGIN

select @num = (max(Номер)+1) from Деталь

END

last\_num\_zapisi Б.33

ALTER PROCEDURE [dbo].[last\_num\_zapisi]

@num int output

AS

BEGIN

select @num = (max(idЗаписи)+1) from Запись

END

list\_of\_days Б.34

ALTER PROCEDURE [dbo].[list\_of\_days]

@name varchar(50)

AS

BEGIN

select Смена.ФИОраб, Смена.ДатаСмены, День.ЧасыРаботы from Смена

join День on День.Дата = Смена.ДатаСмены

where Смена.ФИОраб = @name

END

Mechanic\_list\_of\_cars Б.35

ALTER PROCEDURE [dbo].[Mechanic\_list\_of\_cars]

AS

BEGIN

select \* from ТранспортноеСредство

END

Mechanic\_list\_of\_orders Б.36

ALTER PROCEDURE [dbo].[Mechanic\_list\_of\_orders]

@name varchar(50)

AS

BEGIN

select Номер, Механик, WINномер, СостояниеЗаказа, Стоимость, ДатаОформления,ДатаОкончания from ЗаказНаряд

where Механик = @name and СостояниеЗаказа <> 'Завершён'

END

Mechanic\_update\_count\_detali Б.37

ALTER PROCEDURE [dbo].[Mechanic\_update\_count\_detali]

@num int, @count int

AS

BEGIN

update Деталь set Количество = Количество + @count where Номер = @num

END

Архив Б.38

ALTER PROCEDURE [dbo].[Архив]

@name varchar(50)

AS

BEGIN

select Номер, WINномер, СостояниеЗаказа, Стоимость, ДатаОформления,ДатаОкончания from ЗаказНаряд

where Механик = @name and СостояниеЗаказа = 'Завершён'

END

Триггеры:

Check\_zakaz\_insert Б.39

ALTER TRIGGER [dbo].[Check\_zakaz\_insert]

ON [dbo].[ЗаказНаряд]

AFTER insert

AS

BEGIN

declare @win varchar(50), @date date

select @win = WINномер, @date = ДатаОформления from inserted

if(DATEADD(day, 1, @date) <= getdate() or (@win not in (select WINномер from ТранспортноеСредство)))

begin

rollback tran

print('[Check\_zakaz\_insert]')

end

END

Check\_zakaz\_update Б.40

ALTER TRIGGER [dbo].[Check\_zakaz\_update]

ON [dbo].[ЗаказНаряд]

AFTER UPDATE

AS

BEGIN

declare @date date, @name varchar(50), @sost varchar(50), @date\_end date

select @name=Механик, @sost = СостояниеЗаказа, @date = ДатаОформления from inserted

select @date\_end = ДатаОформления from deleted

if((@date\_end <> @date) or (@name not in (select ФИО from Работник where Должность = 'Механик')) or (@sost not in ('Создан','В процессе','Завершён')))

begin

print('Check\_zakaz\_update')

rollback tran

end

END

Check\_service Б.42

ALTER TRIGGER [dbo].[Check\_service]

ON [dbo].[УслугиЗН]

instead of insert

AS

BEGIN

declare @num int, @service varchar(50)

select @num = НомерНаряда, @service = НазваниеУслуги from inserted

if ((@num in (select НомерНаряда from УслугиЗН where НазваниеУслуги = @service)))

rollback tran

print('[Check\_service]')

print('no')

end

else

begin

insert into УслугиЗН values (@num,'Диагностика')

end

END

Check\_count\_detali Б.43

ALTER TRIGGER [dbo].[Check\_count\_detali]

ON [dbo].[Деталь]

AFTER update

AS

BEGIN

declare @count int, @num int, @count\_delete int

select @num=номер, @count = Количество from inserted

select @count\_delete = Количество from deleted

if (@count < 0)

begin

rollback tran

print('Деталей недостаточно!')

print('Check\_count\_detali')

end

else if (@count\_delete + @count < @count\_delete)

begin

print('Количество заказанных деталей не может быть отрицательным')

rollback tran

end

END

Check\_win Б.43

ALTER TRIGGER [dbo].[Check\_win]

ON [dbo].[ТранспортноеСредство]

AFTER insert

AS

BEGIN

declare @fio varchar(50), @win varchar(50)

select @fio = ФИОклиента, @win=WINномер from inserted

if ((@fio not in (select ФИО from Клиент)))

begin

rollback tran

print('Такой WIN номер существует')

print('[Check\_win]')

end

END

Check\_sotr\_update Б.44

ALTER TRIGGER [dbo].[Check\_sotr\_update]

ON [dbo].[Работник]

AFTER update

AS

BEGIN

declare @fio varchar(50), @age date, @pos varchar(50), @salary money, @stage int, @teleph bigint

select @fio = ФИО, @age = ДатаРождения, @pos = Должность, @salary = Зарплата, @stage = Стаж, @teleph = Телефон from inserted

print(@fio)

if((CONVERT(int,ROUND(DATEDIFF(hour,@age,GETDATE())/8766.0,0))<18) or (@pos not in ('Механик','Менеджер','Директор')) or (@salary < 0) or (@teleph < 0))

begin

rollback tran

print('[Check\_sotr\_update]')

end

END

Check\_update Б.45

ALTER TRIGGER [dbo].[Check\_update]

ON [dbo].[Клиент]

AFTER update

AS

BEGIN

declare @age date, @gender char(1), @teleph bigint

select @age = ДатаРождения, @gender = пол, @teleph = Телефон from inserted

if((CONVERT(int,ROUND(DATEDIFF(hour,@age,GETDATE())/8766.0,0))<18) or (@gender not in ('м','ж')) or (@teleph <= 0))

begin

rollback tran

print('[Check\_update]')

end

END

Check\_insert Б.46

ALTER TRIGGER [dbo].[Check\_insert]

ON [dbo].[Запись]

AFTER insert

AS

BEGIN

declare @fio varchar(50), @date date

select @fio = ФИОклиент, @date = Дата from inserted

if ((@date not in (select Дата from День))or(@fio not in (select ФИО from Клиент)))

begin

rollback tran

print('Check\_insert 1')

end

if (DATEADD(day, 1, @date) <= getdate())

begin

rollback tran

print('Check\_insert 2')

end

END

Check\_client\_age Б.47

ALTER TRIGGER [dbo].[Check\_client\_age]

ON [dbo].[Клиент]

AFTER insert

AS

BEGIN

declare @age date

select @age = ДатаРождения from inserted

if(CONVERT(int,ROUND(DATEDIFF(hour,@age,GETDATE())/8766.0,0))<18)

begin

rollback tran

print('[Check\_client\_age]')

end

END

Check\_complex Б.48

ALTER TRIGGER [dbo].[Check\_complex]

ON [dbo].[Клиент]

AFTER insert

AS

BEGIN

declare @gender char(1), @teleph bigint

select @gender = пол, @teleph = Телефон from inserted

if ((@gender not in ('м','ж')) or @teleph <= 0)

begin

rollback tran

print('[Check\_complex]')

end

END

Check\_sotr\_complex Б.49

ALTER TRIGGER [dbo].[Check\_sotr\_complex]

ON [dbo].[Работник]

AFTER insert

AS

BEGIN

declare @fio varchar(50), @age date, @pos varchar(50), @salary money, @stage int, @teleph bigint

select @fio = ФИО, @age = ДатаРождения, @pos = Должность, @salary = Зарплата, @stage = Стаж, @teleph = Телефон from inserted

print(@fio)

if((CONVERT(int,ROUND(DATEDIFF(hour,@age,GETDATE())/8766.0,0))<18) or (@pos not in ('Механик','Менеджер','Директор')) or (@salary < 0) or (@teleph < 0))

begin

rollback tran

print('[Check\_sotr\_complex]')

end

END

Check\_cost\_service Б.50

ALTER TRIGGER [dbo].[Check\_cost\_service]

ON [dbo].[Услуга]

AFTER insert

AS

BEGIN

declare @cost money

select @cost = Стоимость from inserted

if (@cost <= 0)

begin

rollback tran

print('[Check\_cost\_service]')

end

END

Check\_cost\_update\_service Б.51

ALTER TRIGGER [dbo].[Check\_cost\_update\_service]

ON [dbo].[Услуга]

AFTER update

AS

BEGIN

declare @cost money

select @cost = Стоимость from inserted

if(@cost <= 0)

begin

rollback tran

print('[Check\_cost\_update\_service]')

end

END

Check\_count\_detali Б.52

ALTER TRIGGER [dbo].[Check\_count\_detali]

ON [dbo].[Деталь]

AFTER update

AS

BEGIN

declare @count int, @num int, @count\_delete int

select @num=номер, @count = Количество from inserted

select @count\_delete = Количество from deleted

if (@count < 0)

begin

rollback tran

print('Деталей недостаточно!')

print('Check\_count\_detali')

end

else if (@count\_delete + @count < @count\_delete)

begin

print('Количество заказанных деталей не может быть отрицательным')

rollback tran

end

END

Check\_cost\_update Б.53

ALTER TRIGGER [dbo].[Check\_cost\_update]

ON [dbo].[Деталь]

AFTER update

AS

BEGIN

declare @cost money

select @cost = Стоимость from inserted

if(@cost <= 0)

begin

rollback tran

print('Check\_cost\_update')

end

END

Check\_cost\_count\_detali Б.54

ALTER TRIGGER [dbo].[Check\_cost\_count\_detali]

ON [dbo].[Деталь]

AFTER insert

AS

BEGIN

declare @cost money, @count int

select @cost = стоимость, @count = Количество from inserted

if (@cost <= 0 or @count <= 0)

begin

print('Check\_cost\_count\_detali')

rollback tran

end

END

Check\_pos\_sotr Б.55

ALTER TRIGGER [dbo].[Check\_pos\_sotr]

ON [dbo].[Работник]

AFTER update

AS

BEGIN

declare @pos varchar(50)

select @pos = Должность from inserted

if(@pos not in ('Менеджер','Директор','Механик'))

begin

rollback tran

print('[Check\_pos\_sotr]')

end

END

Check\_sort\_salary Б.56

ALTER TRIGGER [dbo].[Check\_sort\_salary]

ON [dbo].[Работник]

AFTER insert, update

AS

BEGIN

declare @salary money

select @salary = Зарплата from inserted

if (@salary <= 0)

begin

rollback tran

print('[Check\_sort\_salary]')

end

END

Check\_count\_detalii Б.57

ALTER TRIGGER [dbo].[Check\_count\_detalii]

ON [dbo].[Расход]

AFTER insert

AS

BEGIN

declare @count int, @num int

select @num=НомерДетали, @count = Количество from inserted

if (@count > (select Количество from Деталь where Номер = @num))

begin

rollback tran

print('Деталей недостаточно!')

print('Check\_count\_detalii - расход')

end

END

Check\_service Б.58

ALTER TRIGGER [dbo].[Check\_service]

ON [dbo].[УслугиЗН]

instead of insert

AS

BEGIN

declare @num int, @service varchar(50)

select @num = НомерНаряда, @service = НазваниеУслуги from inserted

if ((@num in (select НомерНаряда from УслугиЗН where НазваниеУслуги = @service))) --and (@service in (select НазваниеУслуги from УслугиЗН where НомерНаряда = @num)))

begin

rollback tran

print('[Check\_service]')

print('no')

end

else

begin

insert into УслугиЗН values (@num,'Диагностика')

end

END

Хранимые процедуры дополнение:

admin\_edit\_detali Б.59

ALTER PROCEDURE [dbo].[Admin\_edit\_detali]

@num int, @cost money

AS

BEGIN

update Деталь set Стоимость = @cost where Номер = @num

END

Mechanic\_update\_count\_detali Б.60

ALTER PROCEDURE [dbo].[Mechanic\_update\_count\_detali]

@num int, @count int

AS

BEGIN

update Деталь set Количество = Количество + @count where Номер = @num

END

Admin\_add\_new\_sort Б.61

ALTER PROCEDURE [dbo].[Admin\_add\_new\_sort]

@fio varchar(50), @date date, @state varchar(50), @salary money, @stage int, @teleph bigint, @login varchar(10), @pass varchar(10)

AS

BEGIN

insert into Работник values(@fio, @date, @state, @salary, @stage, @teleph, @login, @pass)

END

Admin\_list\_of\_workers Б.62

ALTER PROCEDURE [dbo].[Admin\_list\_of\_workers]

AS

BEGIN

select \* from Работник order by Должность, ФИО

END

Приложение B. Код программы

Форма Registration.cs B.1

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.SqlClient;

using System.Globalization;

namespace КурсоваяБД

{

public partial class Registration : Form

{

public Registration()

{

InitializeComponent();

this.passField.AutoSize = false;

this.passField.Size = new Size(this.passField.Size.Width, 45);

PositionField.Text = "Должность";

BirthdayField.Text = "ДР";

FioField.Text = "Введите ФИО";

loginField.Text = "Логин";

passField.Text = "Пароль";

FioField.ForeColor = Color.LightGray;

BirthdayField.ForeColor = Color.LightGray;

PositionField.ForeColor = Color.LightGray;

loginField.ForeColor = Color.LightGray;

passField.ForeColor = Color.LightGray;

}

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

{

this.Hide();

LoginForm.Current.Show();

}

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

{

this.Close();

LoginForm.Current.Close();

}

Point lastPoint;

private void label1\_MouseMove(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

this.Left += e.X - lastPoint.X;

this.Top += e.Y - lastPoint.Y;

}

}

private void label1\_MouseDown(object sender, MouseEventArgs e)

{

lastPoint = new Point(e.X, e.Y);

}

private void panel2\_MouseMove(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

this.Left += e.X - lastPoint.X;

this.Top += e.Y - lastPoint.Y;

}

}

private void panel2\_MouseDown(object sender, MouseEventArgs e)

{

lastPoint = new Point(e.X, e.Y);

}

private void FioField\_Enter(object sender, EventArgs e)

{

if (FioField.Text == "Введите ФИО")

{

FioField.Text = "";

FioField.ForeColor = Color.Black;

}

}

private void FioField\_Leave(object sender, EventArgs e)

{

if (FioField.Text == "")

{

FioField.Text = "Введите ФИО";

FioField.ForeColor = Color.LightGray;

}

}

private void BirthdayField\_Enter(object sender, EventArgs e)

{

if (BirthdayField.Text == "ДР")

{

BirthdayField.Text = "";

BirthdayField.ForeColor = Color.Black;

}

}

private void BirthdayField\_Leave(object sender, EventArgs e)

{

if (BirthdayField.Text == "")

{

BirthdayField.Text = "ДР";

BirthdayField.ForeColor = Color.LightGray;

}

}

private void PositionField\_Enter(object sender, EventArgs e)

{

if (PositionField.Text == "Должность")

{

PositionField.Text = "";

PositionField.ForeColor = Color.Black;

}

}

private void PositionField\_Leave(object sender, EventArgs e)

{

if (PositionField.Text == "")

{

PositionField.Text = "Должность";

PositionField.ForeColor = Color.LightGray;

}

}

private void loginField\_Enter(object sender, EventArgs e)

{

if (loginField.Text == "Логин")

{

loginField.Text = "";

loginField.ForeColor = Color.Black;

}

}

private void loginField\_Leave(object sender, EventArgs e)

{

if (loginField.Text == "")

{

loginField.Text = "Логин";

loginField.ForeColor = Color.LightGray;

}

}

private void passField\_Enter(object sender, EventArgs e)

{

if (passField.Text == "Пароль")

{

passField.Text = "";

passField.ForeColor = Color.Black;

}

}

private void passField\_Leave(object sender, EventArgs e)

{

if (passField.Text == "")

{

passField.Text = "Пароль";

passField.ForeColor = Color.LightGray;

}

}

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

{

if (FioField.Text == "Введите ФИО" || passField.Text == "Пароль" || loginField.Text == "Логин" || PositionField.Text == "Должность" || BirthdayField.Text == "ДР")

{

MessageBox.Show("Проверьте поля ввода");

return;

}

if (isUserExists())

return;

if (isPositionCorrect())

return;

if (isDateCorrect())

return;

DB db = new DB();

SqlCommand command = new SqlCommand("Insert into Работник (ФИО,ДатаРождения,Должность,Логин,Пароль) values (@fio,@birth,@pos,@login,@pass)", db.getConnection());

command.Parameters.Add("@fio", SqlDbType.VarChar).Value = FioField.Text;

command.Parameters.Add("@birth", SqlDbType.Date).Value = BirthdayField.Text;

command.Parameters.Add("@pos", SqlDbType.VarChar).Value = PositionField.Text;

command.Parameters.Add("@login", SqlDbType.VarChar).Value = loginField.Text;

command.Parameters.Add("@pass", SqlDbType.VarChar).Value = passField.Text;

db.openConnection();

if (command.ExecuteNonQuery() == 1)

{

MessageBox.Show("Аккаунт был создан");

FioField.Text = "";

}

else

MessageBox.Show("Аккаунт не был создан");

db.closeConnection();

}

static Boolean TextIsDate(string text)

{

var dateFormat = "dd.MM.yyyy";

DateTime scheduleDate;

if (DateTime.TryParseExact(text, dateFormat, DateTimeFormatInfo.InvariantInfo, DateTimeStyles.None, out scheduleDate))

{

return true;

}

return false;

}

public Boolean isDateCorrect()

{

if (TextIsDate(BirthdayField.Text))

{

return false;

}

else

{

MessageBox.Show("Текст не является датой");

return true;

}

}

public Boolean isPositionCorrect()

{

if (PositionField.Text == "Директор" || PositionField.Text == "Механик" || PositionField.Text == "Менеджер")

{

return false;

}

else

MessageBox.Show("Должность введена некорректно\n1. Механик\n2. Менеджер\n3. Директор");

return true;

}

public Boolean isUserExists()

{

DB db = new DB();

DataTable table = new DataTable();

SqlDataAdapter adapter = new SqlDataAdapter();

SqlCommand command = new SqlCommand("select \* from Работник Where Логин = @uL or ФИО = @fio", db.getConnection());

command.Parameters.Add("@uL", SqlDbType.VarChar).Value = loginField.Text;

command.Parameters.Add("@fio", SqlDbType.VarChar).Value = FioField.Text;

adapter.SelectCommand = command;

adapter.Fill(table);

if (table.Rows.Count > 0)

{

MessageBox.Show("Такой пользователь уже зарегистрирован");

return true;

}

else

return false;

}

}

}

Форма LoginForm.cs B.2

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.SqlClient;

namespace КурсоваяБД

{

public partial class LoginForm : Form

{

public static LoginForm Current;

public LoginForm()

{

Current = this;

InitializeComponent();

this.SetStyle(ControlStyles.SupportsTransparentBackColor, true);

this.passField.AutoSize = false;

this.passField.Size = new Size(this.passField.Size.Width, 45);

loginField.Text = "Логин";

passField.Text = "Пароль";

loginField.ForeColor = Color.LightGray;

passField.ForeColor = Color.LightGray;

}

private void close\_MouseClick(object sender, MouseEventArgs e)

{

this.Close();

}

Point lastPoint;

private void panel2\_MouseMove(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

this.Left += e.X - lastPoint.X;

this.Top += e.Y - lastPoint.Y;

}

}

private void panel2\_MouseDown(object sender, MouseEventArgs e)

{

lastPoint = new Point(e.X, e.Y);

}

private void label1\_MouseMove(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

this.Left += e.X - lastPoint.X;

this.Top += e.Y - lastPoint.Y;

}

}

private void label1\_MouseDown(object sender, MouseEventArgs e)

{

lastPoint = new Point(e.X, e.Y);

}

private void pictureBox4\_Click(object sender, EventArgs e)

{

string loginUser = loginField.Text;

string passUser = passField.Text;

DB db = new DB();

DataTable table = new DataTable();

SqlDataAdapter adapter = new SqlDataAdapter();

SqlCommand command = new SqlCommand("select \* from Работник Where Логин = @uL and Пароль = @uP", db.getConnection());

command.Parameters.Add("@uL", SqlDbType.VarChar).Value = loginUser;

command.Parameters.Add("@uP", SqlDbType.VarChar).Value = passUser;

adapter.SelectCommand = command;

adapter.Fill(table);

if (table.Rows.Count == 1 && table.Rows[0][2].ToString() == "Директор")

{

//MessageBox.Show("Авторизация прошла успешно");

Admin admin = new Admin();

admin.label1.Text = ("Директор " + table.Rows[0][0].ToString());

admin.Show();

this.Hide();

//Console.Write("\t{0}", table.Rows[0][2]);

//Console.WriteLine();

//Console.Read();

}

else if (table.Rows.Count == 1 && table.Rows[0][2].ToString() == "Механик")

{

//MessageBox.Show("Авторизация прошла успешно");

Mechanic Mech = new Mechanic();

Mech.label1.Text = ("Механик " + table.Rows[0][0].ToString());

Mech.Show();

this.Hide();

}

else if (table.Rows.Count == 1 && table.Rows[0][2].ToString() == "Менеджер")

{

//MessageBox.Show("Авторизация прошла успешно");

Manager manager = new Manager();

manager.label1.Text = ("Менеджер " + table.Rows[0][0].ToString());

manager.Show();

this.Hide();

}

else

{

MessageBox.Show("Данные введены неверно");

}

}

private void pictureBox3\_Click(object sender, EventArgs e)

{

this.Hide();

Registration registration = new Registration();

registration.Show();

}

private void loginField\_Enter(object sender, EventArgs e)

{

if (loginField.Text == "Логин")

{

loginField.Text = "";

loginField.ForeColor = Color.Black;

}

}

private void loginField\_Leave(object sender, EventArgs e)

{

if (loginField.Text == "")

{

loginField.Text = "Логин";

loginField.ForeColor = Color.LightGray;

}

}

private void passField\_Enter(object sender, EventArgs e)

{

if (passField.Text == "Пароль")

{

passField.Text = "";

passField.ForeColor = Color.Black;

}

}

private void passField\_Leave(object sender, EventArgs e)

{

if (passField.Text == "")

{

passField.Text = "Пароль";

passField.ForeColor = Color.LightGray;

}

}

}

}

Account\_form B.3

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace КурсоваяБД

{

public partial class Account\_form : Form

{

public Account\_form()

{

InitializeComponent();

}

private void Account\_form\_Load(object sender, EventArgs e)

{

this.account\_dataTableAdapter.Fill(this.autoserviceDataSet.account\_data, FIOField.Text);

this.list\_of\_daysTableAdapter.Fill(this.autoserviceDataSet.list\_of\_days, FIOField.Text);

}

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

{

this.Close();

}

private void button1\_Click(object sender, EventArgs e)

{

try

{

DB db = new DB();

db.openConnection();

SqlCommand command = new SqlCommand();

command.CommandType = System.Data.CommandType.StoredProcedure;

command.CommandText = "account\_edit\_data";

command.Connection = db.getConnection();

command.Parameters.Add("@fio", SqlDbType.VarChar).Value = FIOField.Text;

command.Parameters.Add("@date", SqlDbType.DateTime).Value = датаРожденияDateTimePicker.Text;

command.Parameters.Add("@teleph", SqlDbType.BigInt).Value = телефонTextBox.Text;

command.Parameters.Add("@login", SqlDbType.VarChar).Value = логинTextBox.Text;

command.Parameters.Add("@pass", SqlDbType.VarChar).Value = парольTextBox.Text;

command.ExecuteNonQuery();

db.closeConnection();

this.account\_dataTableAdapter.Fill(this.autoserviceDataSet.account\_data, FIOField.Text);

MessageBox.Show("Изминения выполнены успешно!");

}

catch

{

MessageBox.Show("Что-то пошло не так, проверьте что:\n\n 1. Вы совершеннолетний\n 2. Логин не совпадает с другими\n 3. Телефон введен корректно");

}

}

/\*private void button2\_Click(object sender, EventArgs e)

{

list\_of\_daysBindingSource.Filter = "ДатаСмены Like '" + DateField.Text + "%'";

}

private void button3\_Click(object sender, EventArgs e)

{

list\_of\_daysBindingSource.Filter = "";

}\*/

}

}

Add\_detali\_form B.4

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace КурсоваяБД

{

public partial class Add\_detali\_form : Form

{

public static Add\_detali\_form Current;

public Add\_detali\_form()

{

InitializeComponent();

}

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

{

this.Close();

}

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

{

try

{

DB db = new DB();

db.openConnection();

SqlCommand command = new SqlCommand();

command.CommandType = System.Data.CommandType.StoredProcedure;

command.CommandText = "Add\_detali\_to\_zakaz";

command.Connection = db.getConnection();

command.Parameters.Add("@discription", SqlDbType.VarChar).Value = comboBox1.Text;

command.Parameters.Add("@fab", SqlDbType.VarChar).Value = comboBox2.Text;

command.Parameters.Add("@series", SqlDbType.VarChar).Value = comboBox3.Text;

command.Parameters.Add("@count", SqlDbType.Int).Value = CountField.Text;

command.Parameters.Add("@format", SqlDbType.VarChar).Value = FormatField.Text;

command.Parameters.Add("@num\_zakaz", SqlDbType.Int).Value = Mechanik\_edit\_form.Current.номерTextBox.Text;

command.ExecuteNonQuery();

db.closeConnection();

try

{

Mechanik\_edit\_form.Current.list\_of\_detaliTableAdapter.Fill(Mechanik\_edit\_form.Current.autoserviceDataSet.list\_of\_detali, new System.Nullable<int>(((int)(System.Convert.ChangeType(Mechanik\_edit\_form.Current.номерTextBox.Text, typeof(int))))));

}

catch (System.Exception ex)

{

System.Windows.Forms.MessageBox.Show(ex.Message);

}

MessageBox.Show("Деталь успешно добавлена!");

}

catch

{

MessageBox.Show("Что-то пошло не так, проверьте что:\n\n 1. Нужных запчастей на складе достаточно\n 2. Возможно деталь уже есть в заказе");

}

}

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

DB db = new DB();

SqlDataAdapter adapter = new SqlDataAdapter();

SqlCommand command1 = new SqlCommand("select Производитель from Деталь where Описание = @discription group by Производитель", db.getConnection());

command1.Parameters.Add("@discription", SqlDbType.VarChar).Value = comboBox1.Text;

DataTable table1 = new DataTable();

adapter.SelectCommand = command1;

adapter.Fill(table1);

comboBox2.DataSource = table1;

comboBox2.DisplayMember = "Производитель";

SqlCommand command2 = new SqlCommand("select [Серийный номер] from Деталь where Описание = @discription and Производитель = @fab", db.getConnection());

command2.Parameters.Add("@discription", SqlDbType.VarChar).Value = comboBox1.Text;

command2.Parameters.Add("@fab", SqlDbType.VarChar).Value = comboBox2.Text;

DataTable table2 = new DataTable();

adapter.SelectCommand = command2;

adapter.Fill(table2);

comboBox3.DataSource = table2;

comboBox3.DisplayMember = "Серийный номер";

SqlCommand command3 = new SqlCommand("select ЕдИзм from Деталь where Описание = @discription and Производитель = @fab and [Серийный номер] = @series", db.getConnection());

command3.Parameters.Add("@discription", SqlDbType.VarChar).Value = comboBox1.Text;

command3.Parameters.Add("@fab", SqlDbType.VarChar).Value = comboBox2.Text;

command3.Parameters.Add("@series", SqlDbType.VarChar).Value = comboBox3.Text;

DataTable table3 = new DataTable();

adapter.SelectCommand = command3;

adapter.Fill(table3);

FormatField.Text = table3.Rows[0]["ЕдИзм"].ToString();

}

private void comboBox2\_SelectedIndexChanged(object sender, EventArgs e)

{

DB db = new DB();

SqlDataAdapter adapter = new SqlDataAdapter();

SqlCommand command2 = new SqlCommand("select [Серийный номер] from Деталь where Описание = @discription and Производитель = @fab", db.getConnection());

command2.Parameters.Add("@discription", SqlDbType.VarChar).Value = comboBox1.Text;

command2.Parameters.Add("@fab", SqlDbType.VarChar).Value = comboBox2.Text;

DataTable table2 = new DataTable();

adapter.SelectCommand = command2;

adapter.Fill(table2);

comboBox3.DataSource = table2;

comboBox3.DisplayMember = "Серийный номер";

}

Point lastPoint;

private void label1\_MouseMove(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

this.Left += e.X - lastPoint.X;

this.Top += e.Y - lastPoint.Y;

}

}

private void label1\_MouseDown(object sender, MouseEventArgs e)

{

lastPoint = new Point(e.X, e.Y);

}

private void panel2\_MouseMove(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

this.Left += e.X - lastPoint.X;

this.Top += e.Y - lastPoint.Y;

}

}

private void panel2\_MouseDown(object sender, MouseEventArgs e)

{

lastPoint = new Point(e.X, e.Y);

}

}

}

Admin B.5

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 КурсоваяБД

{

public partial class Admin : Form

{

public static Admin Current;

public Admin()

{

Current = this;

InitializeComponent();

}

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

{

this.Close();

LoginForm.Current.Close();

}

private void pictureBox3\_Click(object sender, EventArgs e)

{

this.Hide();

LoginForm.Current.Show();

}

Point lastPoint;

private void label1\_MouseMove(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

this.Left += e.X - lastPoint.X;

this.Top += e.Y - lastPoint.Y;

}

}

private void label1\_MouseDown(object sender, MouseEventArgs e)

{

lastPoint = new Point(e.X, e.Y);

}

private void tabPage1\_MouseMove(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

this.Left += e.X - lastPoint.X;

this.Top += e.Y - lastPoint.Y;

}

}

private void tabPage1\_MouseDown(object sender, MouseEventArgs e)

{

lastPoint = new Point(e.X, e.Y);

}

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

{

// TODO: This line of code loads data into the 'autoserviceDataSet.Admin\_list\_of\_detali' table. You can move, or remove it, as needed.

this.admin\_list\_of\_detaliTableAdapter.Fill(this.autoserviceDataSet.Admin\_list\_of\_detali);

// TODO: This line of code loads data into the 'autoserviceDataSet.Admin\_list\_of\_service' table. You can move, or remove it, as needed.

this.admin\_list\_of\_serviceTableAdapter.Fill(this.autoserviceDataSet.Admin\_list\_of\_service);

// TODO: This line of code loads data into the 'autoserviceDataSet.Admin\_list\_of\_workers' table. You can move, or remove it, as needed.

this.admin\_list\_of\_workersTableAdapter.Fill(this.autoserviceDataSet.Admin\_list\_of\_workers);

}

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

{

System.Windows.Forms.DataGridViewColumn Col =

default(System.Windows.Forms.DataGridViewColumn);

switch (listBox1.SelectedIndex)

{

case 0:

Col = dataGridViewTextBoxColumn1;

break;

case 1:

Col = dataGridViewTextBoxColumn2;

break;

case 2:

Col = dataGridViewTextBoxColumn3;

break;

case 3:

Col = dataGridViewTextBoxColumn4;

break;

case 4:

Col = dataGridViewTextBoxColumn5;

break;

}

if (radioButton1.Checked)

admin\_list\_of\_workersDataGridView.Sort(Col, System.ComponentModel.ListSortDirection.Ascending);

else

admin\_list\_of\_workersDataGridView.Sort(Col, System.ComponentModel.ListSortDirection.Descending);

}

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

{

admin\_list\_of\_workersBindingSource.Filter = "ФИО Like '" + FindField.Text + "%'";

}

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

{

admin\_list\_of\_workersBindingSource.Filter = "";

}

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

{

Admin\_add\_sort\_form admin = new Admin\_add\_sort\_form();

admin.Show();

}

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

{

Admin\_add\_service\_form admin = new Admin\_add\_service\_form();

admin.Show();

}

private void SortButton2\_Click(object sender, EventArgs e)

{

System.Windows.Forms.DataGridViewColumn Col =

default(System.Windows.Forms.DataGridViewColumn);

switch (listBox2.SelectedIndex)

{

case 0:

Col = dataGridViewTextBoxColumn9;

break;

case 1:

Col = dataGridViewTextBoxColumn10;

break;

case 2:

Col = dataGridViewTextBoxColumn11;

break;

}

if (radioButton4.Checked)

admin\_list\_of\_serviceDataGridView.Sort(Col, System.ComponentModel.ListSortDirection.Ascending);

else

admin\_list\_of\_serviceDataGridView.Sort(Col, System.ComponentModel.ListSortDirection.Descending);

}

private void FindBotton2\_Click(object sender, EventArgs e)

{

admin\_list\_of\_serviceBindingSource.Filter = "Название Like '" + FindField2.Text + "%'";

}

private void ShowAllButton2\_Click(object sender, EventArgs e)

{

admin\_list\_of\_serviceBindingSource.Filter = "";

}

private void SortButton3\_Click(object sender, EventArgs e)

{

System.Windows.Forms.DataGridViewColumn Col =

default(System.Windows.Forms.DataGridViewColumn);

switch (listBox3.SelectedIndex)

{

case 0:

Col = dataGridViewTextBoxColumn13;

break;

case 1:

Col = dataGridViewTextBoxColumn14;

break;

case 2:

Col = dataGridViewTextBoxColumn15;

break;

case 3:

Col = dataGridViewTextBoxColumn16;

break;

case 4:

Col = dataGridViewTextBoxColumn18;

break;

}

if (radioButton6.Checked)

admin\_list\_of\_detaliDataGridView.Sort(Col, System.ComponentModel.ListSortDirection.Ascending);

else

admin\_list\_of\_detaliDataGridView.Sort(Col, System.ComponentModel.ListSortDirection.Descending);

}

private void FindButton3\_Click(object sender, EventArgs e)

{

admin\_list\_of\_detaliBindingSource.Filter = "Описание Like '" + FindField3.Text + "%'";

}

private void ShowAllButton3\_Click(object sender, EventArgs e)

{

admin\_list\_of\_detaliBindingSource.Filter = "";

}

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

{

Admin\_add\_detali admin = new Admin\_add\_detali();

admin.Show();

}

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

{

Account\_form account = new Account\_form();

string name = this.label1.Text;

name = name.Remove(0, name.IndexOf(' ') + 1);

account.FIOField.Text = name;

account.Show();

}

private void admin\_list\_of\_workersDataGridView\_CellContentDoubleClick(object sender, DataGridViewCellEventArgs e)

{

Admin\_edit\_sort\_form admin = new Admin\_edit\_sort\_form();

if (e.RowIndex >= 0)

{

DataGridViewRow row = this.admin\_list\_of\_workersDataGridView.Rows[e.RowIndex];

admin.FIOField.Text = row.Cells["dataGridViewTextBoxColumn1"].Value.ToString();

admin.BirthField.Text = row.Cells["dataGridViewTextBoxColumn2"].Value.ToString();

admin.StateField.Text = row.Cells["dataGridViewTextBoxColumn3"].Value.ToString();

admin.SalaryField.Text = row.Cells["dataGridViewTextBoxColumn4"].Value.ToString();

admin.StageField.Text = row.Cells["dataGridViewTextBoxColumn5"].Value.ToString();

admin.TelephField.Text = row.Cells["dataGridViewTextBoxColumn6"].Value.ToString();

admin.LoginField.Text = row.Cells["dataGridViewTextBoxColumn7"].Value.ToString();

admin.PassField.Text = row.Cells["dataGridViewTextBoxColumn8"].Value.ToString();

}

admin.Show();

}

private void admin\_list\_of\_serviceDataGridView\_CellContentDoubleClick(object sender, DataGridViewCellEventArgs e)

{

Admin\_edit\_service\_form admin = new Admin\_edit\_service\_form();

if (e.RowIndex >= 0)

{

DataGridViewRow row = this.admin\_list\_of\_serviceDataGridView.Rows[e.RowIndex];

admin.NameField.Text = row.Cells["dataGridViewTextBoxColumn9"].Value.ToString();

admin.DescriptionField.Text = row.Cells["dataGridViewTextBoxColumn10"].Value.ToString();

admin.CostField.Text = row.Cells["dataGridViewTextBoxColumn11"].Value.ToString();

}

admin.Show();

}

private void admin\_list\_of\_detaliDataGridView\_CellContentDoubleClick(object sender, DataGridViewCellEventArgs e)

{

Admin\_edit\_detali\_form admin = new Admin\_edit\_detali\_form();

if (e.RowIndex >= 0)

{

DataGridViewRow row = this.admin\_list\_of\_detaliDataGridView.Rows[e.RowIndex];

admin.NumField.Text = row.Cells["Номер"].Value.ToString();

admin.FabField.Text = row.Cells["dataGridViewTextBoxColumn13"].Value.ToString();

admin.SeriesField.Text = row.Cells["dataGridViewTextBoxColumn14"].Value.ToString();

admin.DescriptionField.Text = row.Cells["dataGridViewTextBoxColumn15"].Value.ToString();

admin.CountField.Text = row.Cells["dataGridViewTextBoxColumn16"].Value.ToString();

admin.FormatField.Text = row.Cells["dataGridViewTextBoxColumn17"].Value.ToString();

admin.CostField.Text = row.Cells["dataGridViewTextBoxColumn18"].Value.ToString();

}

admin.Show();

}

}

}

Add\_service\_form B.6

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace КурсоваяБД

{

public partial class Add\_service\_form : Form

{

public Add\_service\_form()

{

InitializeComponent();

}

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

{

this.Close();

}

private void Add\_service\_form\_Load(object sender, EventArgs e)

{

DB db = new DB();

SqlDataAdapter adapter = new SqlDataAdapter();

SqlCommand command1 = new SqlCommand("select Название from Услуга", db.getConnection());

DataTable table1 = new DataTable();

adapter.SelectCommand = command1;

adapter.Fill(table1);

comboBox1.DataSource = table1;

comboBox1.DisplayMember = "Название";

}

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

{

try

{

DB db = new DB();

db.openConnection();

SqlCommand command = new SqlCommand();

command.CommandType = System.Data.CommandType.StoredProcedure;

command.CommandText = "Add\_service\_to\_zakaz";

command.Connection = db.getConnection();

command.Parameters.Add("@num\_zakaz", SqlDbType.Int).Value = Mechanik\_edit\_form.Current.номерTextBox.Text;

command.Parameters.Add("@name\_service", SqlDbType.VarChar).Value = comboBox1.Text;

command.ExecuteNonQuery();

db.closeConnection();

try

{

Mechanik\_edit\_form.Current.list\_of\_serviceTableAdapter.Fill(Mechanik\_edit\_form.Current.autoserviceDataSet.list\_of\_service, new System.Nullable<int>(((int)(System.Convert.ChangeType(Mechanik\_edit\_form.Current.номерTextBox.Text, typeof(int))))));

}

catch (System.Exception ex)

{

System.Windows.Forms.MessageBox.Show(ex.Message);

}

MessageBox.Show("Услуга успешно добавлена!");

}

catch

{

MessageBox.Show("Такая услуга уже была оказана!");

}

}

Point lastPoint;

private void label6\_MouseMove(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

this.Left += e.X - lastPoint.X;

this.Top += e.Y - lastPoint.Y;

}

}

private void label6\_MouseDown(object sender, MouseEventArgs e)

{

lastPoint = new Point(e.X, e.Y);

}

private void panel2\_MouseMove(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

this.Left += e.X - lastPoint.X;

this.Top += e.Y - lastPoint.Y;

}

}

private void panel2\_MouseDown(object sender, MouseEventArgs e)

{

lastPoint = new Point(e.X, e.Y);

}

}

}

Admin\_add\_detali B.7

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace КурсоваяБД

{

public partial class Admin\_add\_detali : Form

{

public Admin\_add\_detali()

{

InitializeComponent();

}

private void Admin\_add\_detali\_Load(object sender, EventArgs e)

{

DB db = new DB();

db.openConnection();

SqlCommand command = new SqlCommand();

command.CommandType = System.Data.CommandType.StoredProcedure;

command.CommandText = "last\_num\_detali";

command.Connection = db.getConnection();

command.Parameters.Add("@num", SqlDbType.Int);

command.Parameters["@num"].Direction = ParameterDirection.Output;

command.ExecuteNonQuery();

db.closeConnection();

int result = (int)command.Parameters["@num"].Value;

NumField.Text = Convert.ToString(result);

}

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

{

try

{

DB db = new DB();

db.openConnection();

SqlCommand command = new SqlCommand();

command.CommandType = System.Data.CommandType.StoredProcedure;

command.CommandText = "admin\_add\_detali";

command.Connection = db.getConnection();

command.Parameters.Add("@num", SqlDbType.Int).Value = NumField.Text;

command.Parameters.Add("@fab", SqlDbType.VarChar).Value = FabField.Text;

command.Parameters.Add("@series", SqlDbType.VarChar).Value = SeriesField.Text;

command.Parameters.Add("@description", SqlDbType.VarChar).Value = DescriptionField.Text;

command.Parameters.Add("@count", SqlDbType.Int).Value = CountField.Text;

command.Parameters.Add("@EdIzm", SqlDbType.VarChar).Value = EdIzmField.Text;

command.Parameters.Add("@cost", SqlDbType.Money).Value = CostField.Text;

command.ExecuteNonQuery();

db.closeConnection();

Admin.Current.admin\_list\_of\_detaliTableAdapter.Fill(Admin.Current.autoserviceDataSet.Admin\_list\_of\_detali);

MessageBox.Show("Новая деталь добавлена!");

}

catch

{

MessageBox.Show("Что-т пошло не так, проверьте что:\n\n 1. Стоимость больше нуля\n 2. Количество деталей больше нуля");

}

}

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

{

this.Close();

}

}

}

Admin\_add\_service\_form B.8

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace КурсоваяБД

{

public partial class Admin\_add\_service\_form : Form

{

public Admin\_add\_service\_form()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

try

{

DB db = new DB();

db.openConnection();

SqlCommand command = new SqlCommand();

command.CommandType = System.Data.CommandType.StoredProcedure;

command.CommandText = "Admin\_add\_service";

command.Connection = db.getConnection();

command.Parameters.Add("@name", SqlDbType.VarChar).Value = ServiceNameField.Text;

command.Parameters.Add("@description", SqlDbType.VarChar).Value = DescriptionField.Text;

command.Parameters.Add("@cost", SqlDbType.Money).Value = CostField.Text;

command.ExecuteNonQuery();

db.closeConnection();

Admin.Current.admin\_list\_of\_serviceTableAdapter.Fill(Admin.Current.autoserviceDataSet.Admin\_list\_of\_service);

MessageBox.Show("Новая услуга добавлена");

}

catch

{

MessageBox.Show("Что-то пошло не так, проверьте:\n1. Нет ли уже такой услуги\n2. Стоимость меньше нуля");

}

}

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

{

this.Close();

}

}

}

Admin\_add\_sort\_form B.9

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace КурсоваяБД

{

public partial class Admin\_add\_sort\_form : Form

{

public Admin\_add\_sort\_form()

{

InitializeComponent();

}

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

{

this.Close();

}

private void button1\_Click(object sender, EventArgs e)

{

try

{

DB db = new DB();

db.openConnection();

SqlCommand command = new SqlCommand();

command.CommandType = System.Data.CommandType.StoredProcedure;

command.CommandText = "Admin\_add\_new\_sort";

command.Connection = db.getConnection();

command.Parameters.Add("@fio", SqlDbType.VarChar).Value = FIOField.Text;

command.Parameters.Add("@date", SqlDbType.Date).Value = BirthField.Text;

command.Parameters.Add("@state", SqlDbType.VarChar).Value = StateField.Text;

command.Parameters.Add("@salary", SqlDbType.Money).Value = SalaryField.Text;

command.Parameters.Add("@stage", SqlDbType.Int).Value = StageField.Text;

command.Parameters.Add("@teleph", SqlDbType.BigInt).Value = TelephField.Text;

command.Parameters.Add("@login", SqlDbType.VarChar).Value = LoginField.Text;

command.Parameters.Add("@pass", SqlDbType.VarChar).Value = PassField.Text;

command.ExecuteNonQuery();

db.closeConnection();

Admin.Current.admin\_list\_of\_workersTableAdapter.Fill(Admin.Current.autoserviceDataSet.Admin\_list\_of\_workers);

MessageBox.Show("Новый сотрудник добавлен");

}

catch

{

MessageBox.Show("Что-то пошло не так, проверьте, что:\n\n 1.Зарплата не отрицательна\n 2.Возможно такой логин уже существует\n 3.Возраст больше 18 лет\n 4.Должность существует\n 5.Стаж больше нуля\n 6.Введен телефон\n 7.нет однофалецев\n");

}

}

Point lastPoint;

private void label9\_MouseMove(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

this.Left += e.X - lastPoint.X;

this.Top += e.Y - lastPoint.Y;

}

}

private void label9\_MouseDown(object sender, MouseEventArgs e)

{

lastPoint = new Point(e.X, e.Y);

}

private void panel2\_MouseMove(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

this.Left += e.X - lastPoint.X;

this.Top += e.Y - lastPoint.Y;

}

}

private void panel2\_MouseDown(object sender, MouseEventArgs e)

{

lastPoint = new Point(e.X, e.Y);

}

}

}

Admin\_edit\_detali\_form B.10

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace КурсоваяБД

{

public partial class Admin\_edit\_detali\_form : Form

{

public Admin\_edit\_detali\_form()

{

InitializeComponent();

}

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

{

this.Close();

}

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

{

try

{

DB db = new DB();

db.openConnection();

SqlCommand command = new SqlCommand();

command.CommandType = System.Data.CommandType.StoredProcedure;

command.CommandText = "Admin\_edit\_detali";

command.Connection = db.getConnection();

//@num int, @cost money

command.Parameters.Add("@num", SqlDbType.Int).Value = NumField.Text;

command.Parameters.Add("@cost", SqlDbType.Money).Value = CostField.Text;

command.ExecuteNonQuery();

db.closeConnection();

Admin.Current.admin\_list\_of\_detaliTableAdapter.Fill(Admin.Current.autoserviceDataSet.Admin\_list\_of\_detali);

MessageBox.Show("Изминения выполнены успешно!");

}

catch

{

MessageBox.Show("Стоимость запчасти не может быть отрицательной!");

}

}

}

}

Admin\_edit\_service\_form B.11

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace КурсоваяБД

{

public partial class Admin\_edit\_service\_form : Form

{

public Admin\_edit\_service\_form()

{

InitializeComponent();

}

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

{

this.Close();

}

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

{

try

{

DB db = new DB();

db.openConnection();

SqlCommand command = new SqlCommand();

command.CommandType = System.Data.CommandType.StoredProcedure;

command.CommandText = "Admin\_edit\_service";

command.Connection = db.getConnection();

//@name varchar(50), @des varchar(50), @cost money

command.Parameters.Add("@name", SqlDbType.VarChar).Value = NameField.Text;

command.Parameters.Add("@des", SqlDbType.VarChar).Value = DescriptionField.Text;

command.Parameters.Add("@cost", SqlDbType.Money).Value = CostField.Text;

command.ExecuteNonQuery();

db.closeConnection();

Admin.Current.admin\_list\_of\_serviceTableAdapter.Fill(Admin.Current.autoserviceDataSet.Admin\_list\_of\_service);

MessageBox.Show("Изминения выполнены успешно!");

}

catch

{

MessageBox.Show("Стоимость услуги не может быть отрицательной!");

}

}

}

}

Admin\_edit\_sort\_form B.12

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace КурсоваяБД

{

public partial class Admin\_edit\_sort\_form : Form

{

public Admin\_edit\_sort\_form()

{

InitializeComponent();

}

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

{

this.Close();

}

private void button1\_Click(object sender, EventArgs e)

{

try

{

DB db = new DB();

db.openConnection();

SqlCommand command = new SqlCommand();

command.CommandType = System.Data.CommandType.StoredProcedure;

command.CommandText = "Admin\_edit\_sotr";

command.Connection = db.getConnection();

command.Parameters.Add("@fio", SqlDbType.VarChar).Value = FIOField.Text;

command.Parameters.Add("@pos", SqlDbType.VarChar).Value = StateField.Text;

command.Parameters.Add("@salary", SqlDbType.Money).Value = SalaryField.Text;

command.ExecuteNonQuery();

db.closeConnection();

Admin.Current.admin\_list\_of\_workersTableAdapter.Fill(Admin.Current.autoserviceDataSet.Admin\_list\_of\_workers);

MessageBox.Show("Изминения выполнены успешно!");

}

catch

{

MessageBox.Show("Что-то пошло не так, проверьте что:\n\n 1.Зарплата не отрицательная\n 2.Должность указана верно\n 3.Сотрудник совершеннолетний");

}

}

}

}

Archive\_zakaz\_details\_form B.13

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 КурсоваяБД

{

public partial class Archive\_zakaz\_details\_form : Form

{

public Archive\_zakaz\_details\_form()

{

InitializeComponent();

}

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

{

this.Close();

}

}

}

Manager B.14

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 КурсоваяБД

{

public partial class Manager : Form

{

public static Manager Current;

public Manager()

{

Current = this;

InitializeComponent();

}

private void close\_MouseClick(object sender, MouseEventArgs e)

{

this.Close();

LoginForm.Current.Close();

}

private void pictureBox3\_Click(object sender, EventArgs e)

{

this.Hide();

LoginForm.Current.Show();

}

Point lastPoint;

private void label1\_MouseMove(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

this.Left += e.X - lastPoint.X;

this.Top += e.Y - lastPoint.Y;

}

}

private void label1\_MouseDown(object sender, MouseEventArgs e)

{

lastPoint = new Point(e.X, e.Y);

}

private void tabPage1\_MouseDown(object sender, MouseEventArgs e)

{

lastPoint = new Point(e.X, e.Y);

}

private void tabPage1\_MouseMove(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

this.Left += e.X - lastPoint.X;

this.Top += e.Y - lastPoint.Y;

}

}

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

{

// TODO: This line of code loads data into the 'autoserviceDataSet.Manager\_list\_of\_zapisi\_actual' table. You can move, or remove it, as needed.

this.manager\_list\_of\_zapisi\_actualTableAdapter.Fill(this.autoserviceDataSet.Manager\_list\_of\_zapisi\_actual);

// TODO: This line of code loads data into the 'autoserviceDataSet.Manager\_list\_of\_zapisi' table. You can move, or remove it, as needed.

this.manager\_list\_of\_zapisiTableAdapter.Fill(this.autoserviceDataSet.Manager\_list\_of\_zapisi);

// TODO: This line of code loads data into the 'autoserviceDataSet.Запись' table. You can move, or remove it, as needed.

this.записьTableAdapter.Fill(this.autoserviceDataSet.Запись);

// TODO: This line of code loads data into the 'autoserviceDataSet.Manager\_list\_of\_clients' table. You can move, or remove it, as needed.

this.manager\_list\_of\_clientsTableAdapter.Fill(this.autoserviceDataSet.Manager\_list\_of\_clients);

// TODO: This line of code loads data into the 'autoserviceDataSet.Admin\_list\_of\_service' table. You can move, or remove it, as needed.

this.admin\_list\_of\_serviceTableAdapter.Fill(this.autoserviceDataSet.Admin\_list\_of\_service);

// TODO: This line of code loads data into the 'autoserviceDataSet.Manager\_list\_of\_zakaz' table. You can move, or remove it, as needed.

this.manager\_list\_of\_zakazTableAdapter.Fill(this.autoserviceDataSet.Manager\_list\_of\_zakaz);

}

private void listBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

ConfirmSortButton.Enabled = true;

}

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

{

System.Windows.Forms.DataGridViewColumn Col =

default(System.Windows.Forms.DataGridViewColumn);

switch (listBox1.SelectedIndex)

{

case 0:

Col = dataGridViewTextBoxColumn1;

break;

case 1:

Col = dataGridViewTextBoxColumn2;

break;

case 2:

Col = ФИОклиента;

break;

case 3:

Col = dataGridViewTextBoxColumn4;

break;

case 4:

Col = dataGridViewTextBoxColumn5;

break;

case 5:

Col = dataGridViewTextBoxColumn6;

break;

case 6:

Col = dataGridViewTextBoxColumn7;

break;

}

if (radioButton1.Checked)

manager\_list\_of\_zakazDataGridView.Sort(Col, System.ComponentModel.ListSortDirection.Ascending);

else

manager\_list\_of\_zakazDataGridView.Sort(Col, System.ComponentModel.ListSortDirection.Descending);

} private void FindButton\_Click(object sender, EventArgs e)

{

BindingSource bds = new BindingSource();

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

if (FindField.Text != "")

filterParts.Add("ФИОклиента LIKE '" + FindField.Text + "%'");

if (WINnumField.Text != "")

filterParts.Add("WINномер LIKE '" + WINnumField.Text + "%'");

if (MarkField.Text != "")

filterParts.Add("Модель LIKE '" + MarkField.Text + "%'");

string filter = string.Join(" AND ", filterParts);

manager\_list\_of\_zakazBindingSource.Filter = filter;

}

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

{

manager\_list\_of\_zakazBindingSource.Filter = "";

}

private void SortButton2\_Click(object sender, EventArgs e)

{

System.Windows.Forms.DataGridViewColumn Col =

default(System.Windows.Forms.DataGridViewColumn);

switch (listBox2.SelectedIndex)

{

case 0:

Col = dataGridViewTextBoxColumn9;

break;

case 1:

Col = dataGridViewTextBoxColumn10;

break;

case 2:

Col = dataGridViewTextBoxColumn11;

break;

}

if (radioButton4.Checked)

admin\_list\_of\_serviceDataGridView.Sort(Col, System.ComponentModel.ListSortDirection.Ascending);

else

admin\_list\_of\_serviceDataGridView.Sort(Col, System.ComponentModel.ListSortDirection.Descending);

}

private void FindButton2\_Click(object sender, EventArgs e)

{

admin\_list\_of\_serviceBindingSource.Filter = "Название Like '" + FindServiceField.Text + "%'";

}

private void ShowAllButton2\_Click(object sender, EventArgs e)

{

admin\_list\_of\_serviceBindingSource.Filter = "";

}

private void SortButton3\_Click(object sender, EventArgs e)

{

System.Windows.Forms.DataGridViewColumn Col =

default(System.Windows.Forms.DataGridViewColumn);

switch (listBox3.SelectedIndex)

{

case 0:

Col = dataGridViewTextBoxColumn8;

break;

case 1:

Col = dataGridViewTextBoxColumn12;

break;

case 2:

Col = dataGridViewTextBoxColumn13;

break;

case 3:

Col = dataGridViewTextBoxColumn14;

break;

}

if (radioButton6.Checked)

manager\_list\_of\_clientsDataGridView.Sort(Col, System.ComponentModel.ListSortDirection.Ascending);

else

manager\_list\_of\_clientsDataGridView.Sort(Col, System.ComponentModel.ListSortDirection.Descending);

}

private void FindButton3\_Click(object sender, EventArgs e)

{

manager\_list\_of\_clientsBindingSource.Filter = "ФИО Like '" + FindFioField.Text + "%'";

}

private void ShowAllButton3\_Click(object sender, EventArgs e)

{

admin\_list\_of\_serviceBindingSource.Filter = "";

}

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

{

Manager\_add\_client\_form manager = new Manager\_add\_client\_form();

manager.Show();

}

private void manager\_list\_of\_zakazDataGridView\_CellContentDoubleClick(object sender, DataGridViewCellEventArgs e)

{

Manager\_zakaz\_details\_form manager = new Manager\_zakaz\_details\_form();

if (e.RowIndex >= 0)

{

DataGridViewRow row = this.manager\_list\_of\_zakazDataGridView.Rows[e.RowIndex];

manager.NumField.Text = row.Cells["dataGridViewTextBoxColumn1"].Value.ToString();

}

try

{

manager.list\_of\_detaliTableAdapter.Fill(manager.autoserviceDataSet.list\_of\_detali, new System.Nullable<int>(((int)(System.Convert.ChangeType(manager.NumField.Text, typeof(int))))));

manager.list\_of\_serviceTableAdapter.Fill(manager.autoserviceDataSet.list\_of\_service, new System.Nullable<int>(((int)(System.Convert.ChangeType(manager.NumField.Text, typeof(int))))));

}

catch (System.Exception ex)

{

System.Windows.Forms.MessageBox.Show(ex.Message);

}

manager.Show();

}

private void manager\_list\_of\_clientsDataGridView\_CellContentDoubleClick(object sender, DataGridViewCellEventArgs e)

{

Manager\_edit\_client\_form manager = new Manager\_edit\_client\_form();

if (e.RowIndex >= 0)

{

DataGridViewRow row = this.manager\_list\_of\_clientsDataGridView.Rows[e.RowIndex];

manager.FIOField.Text = row.Cells["dataGridViewTextBoxColumn8"].Value.ToString();

manager.DateField.Text = row.Cells["dataGridViewTextBoxColumn12"].Value.ToString();

manager.GenderField.Text = row.Cells["dataGridViewTextBoxColumn13"].Value.ToString();

manager.TelephField.Text = row.Cells["dataGridViewTextBoxColumn14"].Value.ToString();

}

manager.Show();

}

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

{

Account\_form account = new Account\_form();

string name = this.label1.Text;

name = name.Remove(0, name.IndexOf(' ') + 1);

account.FIOField.Text = name;

account.Show();

}

private void SortButton4\_Click(object sender, EventArgs e)

{

System.Windows.Forms.DataGridViewColumn Col =

default(System.Windows.Forms.DataGridViewColumn);

switch (listBox4.SelectedIndex)

{

case 0:

Col = dataGridViewTextBoxColumn15;

break;

case 1:

Col = dataGridViewTextBoxColumn16;

break;

case 2:

Col = dataGridViewTextBoxColumn17;

break;

}

if (radioButton8.Checked)

manager\_list\_of\_zapisiDataGridView.Sort(Col, System.ComponentModel.ListSortDirection.Ascending);

else

manager\_list\_of\_zapisiDataGridView.Sort(Col, System.ComponentModel.ListSortDirection.Descending);

}

private void FindButton4\_Click(object sender, EventArgs e)

{

manager\_list\_of\_zapisiBindingSource.Filter = "ФИОклиент Like '" + FindField4.Text + "%'"; ;

}

private void ShowAllButton4\_Click(object sender, EventArgs e)

{

manager\_list\_of\_zapisiBindingSource.Filter = "";

}

private void AddButton4\_Click(object sender, EventArgs e)

{

Manager\_add\_zapisi\_form manager = new Manager\_add\_zapisi\_form();

manager.Show();

}

private void manager\_list\_of\_zapisiDataGridView\_CellContentDoubleClick(object sender, DataGridViewCellEventArgs e)

{

Manager\_edit\_or\_delete\_zapisi\_form manager = new Manager\_edit\_or\_delete\_zapisi\_form();

if (e.RowIndex >= 0)

{

DataGridViewRow row = this.manager\_list\_of\_zapisiDataGridView.Rows[e.RowIndex];

manager.NumField.Text = row.Cells["dataGridViewTextBoxColumn15"].Value.ToString();

manager.FioClientField.Text = row.Cells["dataGridViewTextBoxColumn16"].Value.ToString();

manager.dateTimePicker1.Text = row.Cells["dataGridViewTextBoxColumn17"].Value.ToString();

}

manager.Show();

}

}

}

Manager\_add\_client\_form B.15

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace КурсоваяБД

{

public partial class Manager\_add\_client\_form : Form

{

public Manager\_add\_client\_form()

{

InitializeComponent();

}

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

{

//Manager.Current.manager\_list\_of\_clientsTableAdapter.Fill(Manager.Current.autoserviceDataSet.Manager\_list\_of\_clients);

this.Close();

}

private void button1\_Click(object sender, EventArgs e)

{

try

{

DB db = new DB();

db.openConnection();

SqlCommand command = new SqlCommand();

command.CommandType = System.Data.CommandType.StoredProcedure;

command.CommandText = "Add\_new\_client";

command.Connection = db.getConnection();

command.Parameters.Add("@fio", SqlDbType.VarChar).Value = FIOField.Text;

command.Parameters.Add("@date", SqlDbType.Date).Value = BirthField.Text;

command.Parameters.Add("@gender", SqlDbType.Char).Value = GenderField.Text;

command.Parameters.Add("@tel", SqlDbType.BigInt).Value = TelephField.Text;

command.ExecuteNonQuery();

db.closeConnection();

Manager.Current.manager\_list\_of\_clientsTableAdapter.Fill(Manager.Current.autoserviceDataSet.Manager\_list\_of\_clients);

MessageBox.Show("Новый клиент добавлен");

}

catch

{

MessageBox.Show("Что-то пошло не так, проверьте что:\n\n 1.Клиент совершеннолетний\n 2.Телефон введен корретно\n 3.Однофамильцев не существует в базе");

}

}

Point lastPoint;

private void label5\_MouseMove(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

this.Left += e.X - lastPoint.X;

this.Top += e.Y - lastPoint.Y;

}

}

private void label5\_MouseDown(object sender, MouseEventArgs e)

{

lastPoint = new Point(e.X, e.Y);

}

private void panel2\_MouseMove(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

this.Left += e.X - lastPoint.X;

this.Top += e.Y - lastPoint.Y;

}

}

private void panel2\_MouseDown(object sender, MouseEventArgs e)

{

lastPoint = new Point(e.X, e.Y);

}

private void BirthField\_Enter(object sender, EventArgs e)

{

this.BirthField.Format = DateTimePickerFormat.Long;

}

}

}

Manager\_add\_zapisi\_form B.16

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace КурсоваяБД

{

public partial class Manager\_add\_zapisi\_form : Form

{

public Manager\_add\_zapisi\_form()

{

InitializeComponent();

}

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

{

this.Close();

}

private void Manager\_add\_zapisi\_form\_Load(object sender, EventArgs e)

{

DB db = new DB();

db.openConnection();

SqlCommand command = new SqlCommand();

command.CommandType = System.Data.CommandType.StoredProcedure;

command.CommandText = "last\_num\_zapisi";

command.Connection = db.getConnection();

command.Parameters.Add("@num", SqlDbType.Int);

command.Parameters["@num"].Direction = ParameterDirection.Output;

command.ExecuteNonQuery();

db.closeConnection();

int result = (int)command.Parameters["@num"].Value;

NumField.Text = Convert.ToString(result);

SqlDataAdapter adapter = new SqlDataAdapter();

SqlCommand command1 = new SqlCommand("select ФИОклиент from Запись", db.getConnection());

DataTable table1 = new DataTable();

adapter.SelectCommand = command1;

adapter.Fill(table1);

FioClientField.DataSource = table1;

FioClientField.DisplayMember = "ФИОклиент";

}

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

{

try

{

DB db = new DB();

db.openConnection();

SqlCommand command = new SqlCommand();

command.CommandType = System.Data.CommandType.StoredProcedure;

command.CommandText = "Manager\_add\_new\_zapisi";

command.Connection = db.getConnection();

command.Parameters.Add("@num", SqlDbType.Int).Value = NumField.Text;

command.Parameters.Add("@fio", SqlDbType.VarChar).Value = FioClientField.Text;

command.Parameters.Add("@date", SqlDbType.VarChar).Value = dateTimePicker1.Text;

command.ExecuteNonQuery();

db.closeConnection();

Manager.Current.manager\_list\_of\_zapisiTableAdapter.Fill(Manager.Current.autoserviceDataSet.Manager\_list\_of\_zapisi);

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

}

catch

{

MessageBox.Show("Что-то пошло не так, проверьте, что:\n\n 1.Такой клиент существует в базе\n 2.Выбранная дата актуальна");

}

DB db1 = new DB();

db1.openConnection();

SqlCommand command1 = new SqlCommand();

command1.CommandType = System.Data.CommandType.StoredProcedure;

command1.CommandText = "last\_num\_zapisi";

command1.Connection = db1.getConnection();

command1.Parameters.Add("@num", SqlDbType.Int);

command1.Parameters["@num"].Direction = ParameterDirection.Output;

command1.ExecuteNonQuery();

db1.closeConnection();

int result = (int)command1.Parameters["@num"].Value;

NumField.Text = Convert.ToString(result);

}

}

}

Manager\_edit\_client\_form B.17

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace КурсоваяБД

{

public partial class Manager\_edit\_client\_form : Form

{

public Manager\_edit\_client\_form()

{

InitializeComponent();

}

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

{

this.Close();

}

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

{

try

{

DB db = new DB();

db.openConnection();

SqlCommand command = new SqlCommand();

command.CommandType = System.Data.CommandType.StoredProcedure;

command.CommandText = "manager\_edit\_client";

command.Connection = db.getConnection();

command.Parameters.Add("@fio", SqlDbType.VarChar).Value = FIOField.Text;

command.Parameters.Add("@date", SqlDbType.Date).Value = DateField.Text;

command.Parameters.Add("@gender", SqlDbType.Char).Value = GenderField.Text;

command.Parameters.Add("@teleph", SqlDbType.BigInt).Value = TelephField.Text;

command.ExecuteNonQuery();

db.closeConnection();

Manager.Current.manager\_list\_of\_clientsTableAdapter.Fill(Manager.Current.autoserviceDataSet.Manager\_list\_of\_clients);

MessageBox.Show("Изминения выполнены успешно!");

}

catch

{

MessageBox.Show("Что-то пошло нет так, проверьте что:\n\n 1.Клиент совершеннолений\n 2.Пол введен корректно\n 3.Телефон введен корректно");

}

}

}

}

Manager\_edit\_or\_delete\_zapisi\_form B.18

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace КурсоваяБД

{

public partial class Manager\_edit\_or\_delete\_zapisi\_form : Form

{

public Manager\_edit\_or\_delete\_zapisi\_form()

{

InitializeComponent();

}

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

{

this.Close();

}

private void button1\_Click(object sender, EventArgs e)

{

try

{

DB db = new DB();

db.openConnection();

SqlCommand command = new SqlCommand();

command.CommandType = System.Data.CommandType.StoredProcedure;

command.CommandText = "Manager\_update\_zapasi";

command.Connection = db.getConnection();

//@num int, @date date

command.Parameters.Add("@num", SqlDbType.Int).Value = NumField.Text;

command.Parameters.Add("@date", SqlDbType.Date).Value = dateTimePicker1.Text;

command.ExecuteNonQuery();

db.closeConnection();

Manager.Current.manager\_list\_of\_zapisiTableAdapter.Fill(Manager.Current.autoserviceDataSet.Manager\_list\_of\_zapisi);

MessageBox.Show("Изминения выполнены успешно!");

}

catch

{

MessageBox.Show("Что - то пошло не так, проверьте, что:\n\n 1.Выбранная дата сущесвует и не занята");

}

}

private void button2\_Click(object sender, EventArgs e)

{

DB db = new DB();

db.openConnection();

SqlCommand command = new SqlCommand();

command.CommandType = System.Data.CommandType.StoredProcedure;

command.CommandText = "Manager\_delete\_zapisi";

command.Connection = db.getConnection();

//@num int

command.Parameters.Add("@num", SqlDbType.Int).Value = NumField.Text;

command.ExecuteNonQuery();

db.closeConnection();

Manager.Current.manager\_list\_of\_zapisiTableAdapter.Fill(Manager.Current.autoserviceDataSet.Manager\_list\_of\_zapisi);

MessageBox.Show("Удаление выполнено успешно!");

this.Close();

}

}

}

Manager\_zakaz\_details\_form B.19

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 КурсоваяБД

{

public partial class Manager\_zakaz\_details\_form : Form

{

public Manager\_zakaz\_details\_form()

{

InitializeComponent();

}

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

{

this.Close();

}

}

}

Mechanic B.20

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace КурсоваяБД

{

public partial class Mechanic : Form

{

public static Mechanic Current;

public Mechanic()

{

Current = this;

InitializeComponent();

}

private void close\_MouseClick(object sender, MouseEventArgs e)

{

this.Close();

LoginForm.Current.Close();

}

private void pictureBox3\_Click(object sender, EventArgs e)

{

this.Hide();

LoginForm.Current.Show();

}

Point lastPoint;

private void label1\_MouseMove(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

this.Left += e.X - lastPoint.X;

this.Top += e.Y - lastPoint.Y;

}

}

private void label1\_MouseDown(object sender, MouseEventArgs e)

{

lastPoint = new Point(e.X, e.Y);

}

private void tabPage1\_MouseMove(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

this.Left += e.X - lastPoint.X;

this.Top += e.Y - lastPoint.Y;

}

}

private void tabPage1\_MouseDown(object sender, MouseEventArgs e)

{

lastPoint = new Point(e.X, e.Y);

}

private void Mechanic\_Load\_1(object sender, EventArgs e)

{

// TODO: This line of code loads data into the 'autoserviceDataSet.Admin\_list\_of\_detali' table. You can move, or remove it, as needed.

this.admin\_list\_of\_detaliTableAdapter.Fill(this.autoserviceDataSet.Admin\_list\_of\_detali);

// TODO: This line of code loads data into the 'autoserviceDataSet.Mechanic\_list\_of\_cars' table. You can move, or remove it, as needed.

this.mechanic\_list\_of\_carsTableAdapter.Fill(this.autoserviceDataSet.Mechanic\_list\_of\_cars);

string name = this.label1.Text;

name = name.Remove(0, name.IndexOf(' ') + 1);

try

{

this.архивTableAdapter.Fill(this.autoserviceDataSet.Архив, name);

this.mechanic\_list\_of\_ordersTableAdapter.Fill(this.autoserviceDataSet.Mechanic\_list\_of\_orders, name);

}

catch (System.Exception ex)

{

System.Windows.Forms.MessageBox.Show(ex.Message);

}

}

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

{

Mechanic\_add\_form mech = new Mechanic\_add\_form();

mech.label1.Text = (this.label1.Text);

//this.Close();

DB db = new DB();

DataTable table = new DataTable();

SqlDataAdapter adapter = new SqlDataAdapter();

SqlCommand command = new SqlCommand("select WINномер from ТранспортноеСредство", db.getConnection());

adapter.SelectCommand = command;

adapter.Fill(table);

mech.wINномерTextBox.DataSource = table;

mech.wINномерTextBox.DisplayMember = "WINномер";

mech.Show();

}

private void mechanic\_list\_of\_ordersDataGridView\_CellContentDoubleClick(object sender, DataGridViewCellEventArgs e)

{

Mechanik\_edit\_form mech = new Mechanik\_edit\_form();

mech.label1.Text = (this.label1.Text);

string name = this.label1.Text;

name = name.Remove(0, name.IndexOf(' ') + 1);

mech.механикTextBox.Text = name;

if (e.RowIndex >= 0)

{

DataGridViewRow row = this.mechanic\_list\_of\_ordersDataGridView.Rows[e.RowIndex];

mech.номерTextBox.Text = row.Cells["Номер"].Value.ToString();

mech.wINномерTextBox.Text = row.Cells["WINномер"].Value.ToString();

mech.СостояниеЗаказаBox1.Text = row.Cells["СостояниеЗаказа"].Value.ToString();

//mech.стоимостьTextBox.Text = row.Cells["Стоимость"].Value.ToString();

mech.датаОформленияDateTimePicker.Text = row.Cells["ДатаОформления"].Value.ToString();

}

int cost = 0;

try

{

mech.list\_of\_detaliTableAdapter.Fill(mech.autoserviceDataSet.list\_of\_detali, new System.Nullable<int>(((int)(System.Convert.ChangeType(mech.номерTextBox.Text, typeof(int))))));

mech.list\_of\_serviceTableAdapter.Fill(mech.autoserviceDataSet.list\_of\_service, new System.Nullable<int>(((int)(System.Convert.ChangeType(mech.номерTextBox.Text, typeof(int))))));

}

catch (System.Exception ex)

{

System.Windows.Forms.MessageBox.Show(ex.Message);

}

for (int i = 0; i < mech.list\_of\_detaliDataGridView.RowCount; i++)

{

cost += (Convert.ToInt32(mech.list\_of\_detaliDataGridView[3, i].Value)\*Convert.ToInt32(mech.list\_of\_detaliDataGridView[4, i].Value));

}

for (int i = 0; i < mech.list\_of\_serviceDataGridView.RowCount; i++)

{

cost += Convert.ToInt32(mech.list\_of\_serviceDataGridView[1, i].Value);

}

mech.стоимостьTextBox.Text = Convert.ToString(cost);

this.Close();

mech.Show();

}

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

{

System.Windows.Forms.DataGridViewColumn Col =

default(System.Windows.Forms.DataGridViewColumn);

switch (listBox.SelectedIndex)

{

case 0:

Col = Номер;

break;

case 1:

Col = WINномер;

break;

case 2:

Col = СостояниеЗаказа;

break;

case 3:

Col = Стоимость;

break;

case 4:

Col = ДатаОформления;

break;

case 5:

Col = ДатаОкончания;

break;

}

if (radioButton4.Checked)

mechanic\_list\_of\_ordersDataGridView.Sort(Col, System.ComponentModel.ListSortDirection.Ascending);

else

mechanic\_list\_of\_ordersDataGridView.Sort(Col, System.ComponentModel.ListSortDirection.Descending);

} private void FindButton\_Click(object sender, EventArgs e)

{

mechanic\_list\_of\_ordersBindingSource.Filter = "WINномер Like '" + FindWINField.Text + "%'";

}

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

{

mechanic\_list\_of\_ordersBindingSource.Filter = "";

}

private void SortButton2\_Click(object sender, EventArgs e)

{

System.Windows.Forms.DataGridViewColumn Col =

default(System.Windows.Forms.DataGridViewColumn);

switch (listBox1.SelectedIndex)

{

case 0:

Col = dataGridViewTextBoxColumn2;

break;

case 1:

Col = dataGridViewTextBoxColumn8;

break;

case 2:

Col = dataGridViewTextBoxColumn9;

break;

case 3:

Col = dataGridViewTextBoxColumn10;

break;

case 4:

Col = dataGridViewTextBoxColumn11;

break;

case 5:

Col = dataGridViewTextBoxColumn12;

break;

}

if (radioButton2.Checked)

архивDataGridView.Sort(Col, System.ComponentModel.ListSortDirection.Ascending);

else

архивDataGridView.Sort(Col, System.ComponentModel.ListSortDirection.Descending);

}

private void FindButton2\_Click(object sender, EventArgs e)

{

архивBindingSource.Filter = "WINномер Like '" + FindField.Text + "%'";

}

private void ShowAllButton2\_Click(object sender, EventArgs e)

{

архивBindingSource.Filter = "";

}

private void архивDataGridView\_CellContentDoubleClick(object sender, DataGridViewCellEventArgs e)

{

Archive\_zakaz\_details\_form mech = new Archive\_zakaz\_details\_form();

if (e.RowIndex >= 0)

{

DataGridViewRow row = this.архивDataGridView.Rows[e.RowIndex];

mech.NumField.Text = row.Cells["dataGridViewTextBoxColumn2"].Value.ToString();

}

try

{

mech.list\_of\_detaliTableAdapter.Fill(mech.autoserviceDataSet.list\_of\_detali, new System.Nullable<int>(((int)(System.Convert.ChangeType(mech.NumField.Text, typeof(int))))));

mech.list\_of\_serviceTableAdapter.Fill(mech.autoserviceDataSet.list\_of\_service, new System.Nullable<int>(((int)(System.Convert.ChangeType(mech.NumField.Text, typeof(int))))));

}

catch (System.Exception ex)

{

System.Windows.Forms.MessageBox.Show(ex.Message);

}

mech.Show();

}

private void SortButton3\_Click(object sender, EventArgs e)

{

System.Windows.Forms.DataGridViewColumn Col =

default(System.Windows.Forms.DataGridViewColumn);

switch (listBox2.SelectedIndex)

{

case 0:

Col = dataGridViewTextBoxColumn1;

break;

case 1:

Col = dataGridViewTextBoxColumn3;

break;

case 2:

Col = dataGridViewTextBoxColumn4;

break;

case 3:

Col = dataGridViewTextBoxColumn5;

break;

case 4:

Col = dataGridViewTextBoxColumn6;

break;

}

if (radioButton6.Checked)

mechanic\_list\_of\_carsDataGridView.Sort(Col, System.ComponentModel.ListSortDirection.Ascending);

else

mechanic\_list\_of\_carsDataGridView.Sort(Col, System.ComponentModel.ListSortDirection.Descending);

}

private void FindButton3\_Click(object sender, EventArgs e)

{

mechanic\_list\_of\_carsBindingSource.Filter = "WINномер Like '" + FindField3.Text + "%'";

}

private void ShowAllButton3\_Click(object sender, EventArgs e)

{

mechanic\_list\_of\_carsBindingSource.Filter = "";

}

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

{

Mechanik\_new\_car\_form mech = new Mechanik\_new\_car\_form();

string name = this.label1.Text;

name = name.Remove(0, name.IndexOf(' ') + 1);

mech.FIOMechField.Text = name;

mech.Show();

}

private void SortButton4\_Click(object sender, EventArgs e)

{

System.Windows.Forms.DataGridViewColumn Col =

default(System.Windows.Forms.DataGridViewColumn);

switch (listBox3.SelectedIndex)

{

case 0:

Col = dataGridViewTextBoxColumn13;

break;

case 1:

Col = dataGridViewTextBoxColumn14;

break;

case 2:

Col = dataGridViewTextBoxColumn15;

break;

case 3:

Col = dataGridViewTextBoxColumn16;

break;

case 4:

Col = dataGridViewTextBoxColumn18;

break;

}

if (radioButton8.Checked)

admin\_list\_of\_detaliDataGridView.Sort(Col, System.ComponentModel.ListSortDirection.Ascending);

else

admin\_list\_of\_detaliDataGridView.Sort(Col, System.ComponentModel.ListSortDirection.Descending);

}

private void FindButton4\_Click(object sender, EventArgs e)

{

admin\_list\_of\_detaliBindingSource.Filter = "Описание Like '" + FindField3.Text + "%'";

}

private void ShowAllButton4\_Click(object sender, EventArgs e)

{

admin\_list\_of\_detaliBindingSource.Filter = "";

}

private void admin\_list\_of\_detaliDataGridView\_CellContentDoubleClick(object sender, DataGridViewCellEventArgs e)

{

Mechanic\_zakaz\_detali\_form mech = new Mechanic\_zakaz\_detali\_form();

if (e.RowIndex >= 0)

{

DataGridViewRow row = this.admin\_list\_of\_detaliDataGridView.Rows[e.RowIndex];

mech.NumField.Text = row.Cells["dataGridViewTextBoxColumn7"].Value.ToString();

mech.TypeField.Text = row.Cells["dataGridViewTextBoxColumn15"].Value.ToString();

mech.FabField.Text = row.Cells["dataGridViewTextBoxColumn13"].Value.ToString();

mech.SeriesField.Text = row.Cells["dataGridViewTextBoxColumn14"].Value.ToString();

mech.FormatField.Text = row.Cells["dataGridViewTextBoxColumn17"].Value.ToString();

}

mech.Show();

}

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

{

Account\_form account = new Account\_form();

string name = this.label1.Text;

name = name.Remove(0, name.IndexOf(' ') + 1);

account.FIOField.Text = name;

account.Show();

}

}

}

Mechanic\_add\_form B.21

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace КурсоваяБД

{

public partial class Mechanic\_add\_form : Form

{

public Mechanic\_add\_form()

{

InitializeComponent();

}

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

{

//Mechanic Mech = new Mechanic();

//Mech.label1.Text = (this.label1.Text);

// Mech.Show();

this.Close();

}

private void Mechanic\_add\_form\_Load(object sender, EventArgs e)

{

string name = this.label1.Text;

name = name.Remove(0, name.IndexOf(' ') + 1);

механикTextBox.Text = name;

DB db = new DB();

db.openConnection();

SqlCommand command = new SqlCommand();

command.CommandType = System.Data.CommandType.StoredProcedure;

command.CommandText = "last\_num";

command.Connection = db.getConnection();

command.Parameters.Add("@num", SqlDbType.Int);

command.Parameters["@num"].Direction = ParameterDirection.Output;

command.ExecuteNonQuery();

db.closeConnection();

int result = (int)command.Parameters["@num"].Value;

номерTextBox.Text = Convert.ToString(result);

//

//

//

db.openConnection();

SqlCommand command2 = new SqlCommand();

command2.CommandType = System.Data.CommandType.StoredProcedure;

command2.CommandText = "diagnostic\_cost";

command2.Connection = db.getConnection();

command2.Parameters.Add("@cost", SqlDbType.Int);

command2.Parameters["@cost"].Direction = ParameterDirection.Output;

command2.ExecuteNonQuery();

db.closeConnection();

int result2 = (int)command2.Parameters["@cost"].Value;

стоимостьTextBox.Text = Convert.ToString(result2);

}

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

{

try

{

DB db = new DB();

db.openConnection();

SqlCommand command = new SqlCommand();

command.CommandType = System.Data.CommandType.StoredProcedure;

command.CommandText = "add\_new\_zakaz";

command.Connection = db.getConnection();

command.Parameters.Add("@num", SqlDbType.Int).Value = номерTextBox.Text;

command.Parameters.Add("@name", SqlDbType.VarChar).Value = механикTextBox.Text;

command.Parameters.Add("@win", SqlDbType.VarChar).Value = wINномерTextBox.Text;

command.Parameters.Add("@sost", SqlDbType.VarChar).Value = состояниеЗаказаTextBox.Text;

command.Parameters.Add("@cost", SqlDbType.Money).Value = стоимостьTextBox.Text;

command.Parameters.Add("@date", SqlDbType.Date).Value = датаОформленияDateTimePicker.Text;

command.ExecuteNonQuery();

db.closeConnection();

Mechanic.Current.mechanic\_list\_of\_ordersTableAdapter.Fill(Mechanic.Current.autoserviceDataSet.Mechanic\_list\_of\_orders, механикTextBox.Text);

MessageBox.Show("Заказ-наряд создан");

}

catch

{

MessageBox.Show("Что-то пошло не так, проверьте что:\n\n 1.Введена корректная дата\n 2.Такой win номер существует");

}

}

private void textBox1\_TextChanged(object sender, EventArgs e)

{

DB db = new DB();

db.openConnection();

SqlCommand command = new SqlCommand();

command.CommandType = System.Data.CommandType.StoredProcedure;

command.CommandText = "filter\_wincode";

command.Connection = db.getConnection();

command.Parameters.Add("@win", SqlDbType.VarChar).Value = FilterField.Text;

DataTable table = new DataTable();

SqlDataAdapter adapter = new SqlDataAdapter();

adapter.SelectCommand = command;

adapter.Fill(table);

wINномерTextBox.DataSource = table;

wINномерTextBox.DisplayMember = "WINномер";

command.ExecuteNonQuery();

db.closeConnection();

}

}

}

Mechanic\_zakaz\_detali\_form B.22

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace КурсоваяБД

{

public partial class Mechanic\_zakaz\_detali\_form : Form

{

public Mechanic\_zakaz\_detali\_form()

{

InitializeComponent();

}

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

{

try

{

DB db = new DB();

db.openConnection();

SqlCommand command = new SqlCommand();

command.CommandType = System.Data.CommandType.StoredProcedure;

command.CommandText = "Mechanic\_update\_count\_detali";

command.Connection = db.getConnection();

command.Parameters.Add("@num", SqlDbType.Int).Value = NumField.Text;

if (CountField.Text == "")

{

MessageBox.Show("Вы не ввели количество");

}

else

{

command.Parameters.Add("@count", SqlDbType.Int).Value = CountField.Text;

}

command.ExecuteNonQuery();

db.closeConnection();

Mechanic.Current.admin\_list\_of\_detaliTableAdapter.Fill(Mechanic.Current.autoserviceDataSet.Admin\_list\_of\_detali);

MessageBox.Show("Детали успешно заказаны");

}

catch

{

MessageBox.Show("Количество не может быть отрицательным!");

}

}

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

{

this.Close();

}

}

}

Mechanik\_edit\_form B.23

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace КурсоваяБД

{

public partial class Mechanik\_edit\_form : Form

{

public static Mechanik\_edit\_form Current;

public Mechanik\_edit\_form()

{

Current = this;

InitializeComponent();

}

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

{

Mechanic Mech = new Mechanic();

Mech.label1.Text = (this.label1.Text);

DB db = new DB();

db.openConnection();

SqlCommand command = new SqlCommand();

command.CommandType = System.Data.CommandType.StoredProcedure;

command.CommandText = "Edit\_zakaz";

command.Connection = db.getConnection();

command.Parameters.Add("@name", SqlDbType.VarChar).Value = механикTextBox.Text;

command.Parameters.Add("@sost", SqlDbType.VarChar).Value = СостояниеЗаказаBox1.Text;

command.Parameters.Add("@cost", SqlDbType.Money).Value = стоимостьTextBox.Text;

if (датаОкончанияDateTimePicker.Text == " ")

{

command.Parameters.Add("@date", SqlDbType.Date).Value = DBNull.Value;

}

else

{

command.Parameters.Add("@date", SqlDbType.Date).Value = датаОкончанияDateTimePicker.Text;

}

command.Parameters.Add("@num", SqlDbType.Int).Value = номерTextBox.Text;

command.ExecuteNonQuery();

db.closeConnection();

Mech.Show();

this.Close();

}

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

{

try

{

DB db = new DB();

db.openConnection();

SqlCommand command = new SqlCommand();

command.CommandType = System.Data.CommandType.StoredProcedure;

command.CommandText = "Edit\_zakaz";

command.Connection = db.getConnection();

command.Parameters.Add("@name", SqlDbType.VarChar).Value = механикTextBox.Text;

command.Parameters.Add("@sost", SqlDbType.VarChar).Value = СостояниеЗаказаBox1.Text;

command.Parameters.Add("@cost", SqlDbType.Money).Value = стоимостьTextBox.Text;

if (датаОкончанияDateTimePicker.Text == " ")

{

command.Parameters.Add("@date", SqlDbType.Date).Value = DBNull.Value;

}

else

{

command.Parameters.Add("@date", SqlDbType.Date).Value = датаОкончанияDateTimePicker.Text;

}

command.Parameters.Add("@num", SqlDbType.Int).Value = номерTextBox.Text;

command.ExecuteNonQuery();

db.closeConnection();

MessageBox.Show("Изминения выполнены успешно!");

}

catch

{

MessageBox.Show("Что-то пошло не так, проверьте что:\n\n 1.ФИО механика указано верно\n 2.Состояние заказа указано верно\n 3.Дата оформления и окончания указаны верно");

}

}

Point lastPoint;

private void panel2\_MouseMove(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

this.Left += e.X - lastPoint.X;

this.Top += e.Y - lastPoint.Y;

}

}

private void panel2\_MouseDown(object sender, MouseEventArgs e)

{

lastPoint = new Point(e.X, e.Y);

}

private void label2\_MouseMove(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

this.Left += e.X - lastPoint.X;

this.Top += e.Y - lastPoint.Y;

}

}

private void label2\_MouseDown(object sender, MouseEventArgs e)

{

lastPoint = new Point(e.X, e.Y);

}

private void датаОкончанияDateTimePicker\_Enter(object sender, EventArgs e)

{

this.датаОкончанияDateTimePicker.Format = DateTimePickerFormat.Long;

}

private void list\_of\_detaliDataGridView\_MouseDoubleClick(object sender, MouseEventArgs e)

{

Add\_detali\_form add = new Add\_detali\_form();

add.Show();

DB db = new DB();

DataTable table = new DataTable();

SqlDataAdapter adapter = new SqlDataAdapter();

SqlCommand command = new SqlCommand("select Описание from Деталь group by Описание", db.getConnection());

adapter.SelectCommand = command;

adapter.Fill(table);

add.comboBox1.DataSource = table;

add.comboBox1.DisplayMember = "Описание";

SqlCommand command1 = new SqlCommand("select Производитель from Деталь where Описание = @discription group by Производитель", db.getConnection());

command1.Parameters.Add("@discription", SqlDbType.VarChar).Value = add.comboBox1.Text;

DataTable table1 = new DataTable();

adapter.SelectCommand = command1;

adapter.Fill(table1);

add.comboBox2.DataSource = table1;

add.comboBox2.DisplayMember = "Производитель";

SqlCommand command2 = new SqlCommand("select [Серийный номер] from Деталь where Описание = @discription and Производитель = @fab", db.getConnection());

command2.Parameters.Add("@discription", SqlDbType.VarChar).Value = add.comboBox1.Text;

command2.Parameters.Add("@fab", SqlDbType.VarChar).Value = add.comboBox2.Text;

DataTable table2 = new DataTable();

adapter.SelectCommand = command2;

adapter.Fill(table2);

add.comboBox3.DataSource = table2;

add.comboBox3.DisplayMember = "Серийный номер";

SqlCommand command3 = new SqlCommand("select ЕдИзм from Деталь where Описание = @discription and Производитель = @fab and [Серийный номер] = @series", db.getConnection());

command3.Parameters.Add("@discription", SqlDbType.VarChar).Value = add.comboBox1.Text;

command3.Parameters.Add("@fab", SqlDbType.VarChar).Value = add.comboBox2.Text;

command3.Parameters.Add("@series", SqlDbType.VarChar).Value = add.comboBox3.Text;

DataTable table3 = new DataTable();

adapter.SelectCommand = command3;

adapter.Fill(table3);

add.FormatField.Text = table3.Rows[0]["ЕдИзм"].ToString();

}

private void list\_of\_serviceDataGridView\_MouseDoubleClick(object sender, MouseEventArgs e)

{

Add\_service\_form add = new Add\_service\_form();

add.Show();

}

private void list\_of\_serviceDataGridView\_RowsAdded(object sender, DataGridViewRowsAddedEventArgs e)

{

int cost = 0;

for (int i = 0; i < list\_of\_detaliDataGridView.RowCount; i++)

{

cost += (Convert.ToInt32(list\_of\_detaliDataGridView[3, i].Value) \* Convert.ToInt32(list\_of\_detaliDataGridView[4, i].Value));

}

for (int i = 0; i < list\_of\_serviceDataGridView.RowCount; i++)

{

cost += Convert.ToInt32(list\_of\_serviceDataGridView[1, i].Value);

}

стоимостьTextBox.Text = Convert.ToString(cost);

}

private void list\_of\_detaliDataGridView\_RowsAdded(object sender, DataGridViewRowsAddedEventArgs e)

{

int cost = 0;

for (int i = 0; i < list\_of\_detaliDataGridView.RowCount; i++)

{

cost += (Convert.ToInt32(list\_of\_detaliDataGridView[3, i].Value) \* Convert.ToInt32(list\_of\_detaliDataGridView[4, i].Value));

}

for (int i = 0; i < list\_of\_serviceDataGridView.RowCount; i++)

{

cost += Convert.ToInt32(list\_of\_serviceDataGridView[1, i].Value);

}

стоимостьTextBox.Text = Convert.ToString(cost);

}

private void СостояниеЗаказаBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

if (СостояниеЗаказаBox1.Text == "Завершён")

{

this.датаОкончанияDateTimePicker.Format = DateTimePickerFormat.Long;

}

else if (СостояниеЗаказаBox1.Text == "В процессе")

{

this.датаОкончанияDateTimePicker.Format = DateTimePickerFormat.Custom;

СостояниеЗаказаBox1.Text = "";

}

}

}

}

Mechanik\_new\_car\_form B.24

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace КурсоваяБД

{

public partial class Mechanik\_new\_car\_form : Form

{

public Mechanik\_new\_car\_form()

{

InitializeComponent();

}

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

{

try

{

DB db = new DB();

db.openConnection();

SqlCommand command = new SqlCommand();

command.CommandType = System.Data.CommandType.StoredProcedure;

command.CommandText = "Mechanic\_new\_car";

command.Connection = db.getConnection();

command.Parameters.Add("@win", SqlDbType.VarChar).Value = WINField.Text;

command.Parameters.Add("@fioMech", SqlDbType.VarChar).Value = FIOMechField.Text;

command.Parameters.Add("@fioClient", SqlDbType.VarChar).Value = FIOClientField.Text;

command.Parameters.Add("@fab", SqlDbType.VarChar).Value = FabField.Text;

command.Parameters.Add("@model", SqlDbType.VarChar).Value = ModelField.Text;

command.ExecuteNonQuery();

db.closeConnection();

Mechanic.Current.mechanic\_list\_of\_carsTableAdapter.Fill(Mechanic.Current.autoserviceDataSet.Mechanic\_list\_of\_cars);

MessageBox.Show("Новое транспортное средство добавлено");

}

catch

{

MessageBox.Show("Что-то пошло не так, проверьте что:\n\n 1.Такой WIN номер прежде не был указан\n 2.Такой клиент есть в базе");

}

}

private void Mechanik\_new\_car\_form\_Load(object sender, EventArgs e)

{

DB db = new DB();

SqlDataAdapter adapter = new SqlDataAdapter();

SqlCommand command1 = new SqlCommand("select ФИО from Клиент where ФИО not in (select ФИОклиента from ТранспортноеСредство)", db.getConnection());

DataTable table1 = new DataTable();

adapter.SelectCommand = command1;

adapter.Fill(table1);

FIOClientField.DataSource = table1;

FIOClientField.DisplayMember = "ФИО";

}

private void NewClientCheck\_CheckedChanged(object sender, EventArgs e)

{

DB db = new DB();

SqlDataAdapter adapter = new SqlDataAdapter();

if (NewClientCheck.Checked)

{

SqlCommand command1 = new SqlCommand("select ФИО from Клиент where ФИО not in (select ФИОклиента from ТранспортноеСредство)", db.getConnection());

DataTable table1 = new DataTable();

adapter.SelectCommand = command1;

adapter.Fill(table1);

FIOClientField.DataSource = table1;

FIOClientField.DisplayMember = "ФИО";

}

else

{

SqlCommand command1 = new SqlCommand("select ФИО from Клиент where ФИО in (select ФИОклиента from ТранспортноеСредство)", db.getConnection());

DataTable table1 = new DataTable();

adapter.SelectCommand = command1;

adapter.Fill(table1);

FIOClientField.DataSource = table1;

FIOClientField.DisplayMember = "ФИО";

}

}

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

{

this.Close();

}

}

}