СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 5](#_Toc99101336)

[1. ИССЛЕДОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ 8](#_Toc99101337)

[2. ПОСТАНОВКА ЗАДАЧИ И ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА 9](#_Toc99101338)

[2.1. Техническое задание 9](#_Toc99101339)

[2.2. Требования к программе 9](#_Toc99101340)

[2.3. Требования к программной документации 10](#_Toc99101341)

[2.4. Стадии и этапы разработки 10](#_Toc99101342)

[2.5. Проектирование UML 10](#_Toc99101343)

[2.6. Функциональная модель 11](#_Toc99101344)

[2.7. Структура базы данных 14](#_Toc99101345)

[3. РЕАЛИЗАЦИЯ. ОПИСАНИЕ АЛГОРИТМОВ РАБОТЫ ПРОГРАММНОГО СРЕДСТВА 16](#_Toc99101346)

[3.1. Алгоритм работы ПО 16](#_Toc99101347)

[3.2. Общие сведения 16](#_Toc99101348)

[3.3. Функциональное назначение 16](#_Toc99101349)

[3.4. Требование к языку в среде разработки 17](#_Toc99101350)

[3.5. Описание таблиц базы данных 19](#_Toc99101351)

[3.7. Описание алгоритма работы с приложением 24](#_Toc99101352)

[3.8. Описание основных модулей программы 24](#_Toc99101353)

[3.9. Проектирование интерфейса программного средства 25](#_Toc99101354)

[4. ТЕСТИРОВАНИЕ 34](#_Toc99101355)

[5. ЭКОНОМИЧЕСКИЙ РАЗДЕЛ 37](#_Toc99101356)

[5.1. Определение себестоимости программного обеспечения (ПО) 37](#_Toc99101357)

[5.2. Расчет затрат на материалы 38](#_Toc99101358)

[5.3. Обоснование трудоемкости разработки ПО. 38](#_Toc99101359)

[5.4. Расчет стоимости машинного времени 39](#_Toc99101360)

[5.5. Расчет затрат на оплату труда 40](#_Toc99101361)

[5.6. Расчет полной себестоимости программного обеспечения 41](#_Toc99101362)

[ЗАКЛЮЧЕНИЕ 43](#_Toc99101376)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 44](#_Toc99101377)

[ПРИЛОЖЕНИЕ 1](#_Toc99101378). [ИСХОДНЫЙ ТЕКСТ ПРОГРАММНОГО СРЕДСТВА 45](#_Toc99101379)

[ПРИЛОЖЕНИЕ 2](#_Toc99101380). [БЛОК-СХЕМА АЛГОРИТМА РАБОТЫ ПРОГРАММНОГО СРЕДСТВА 77](#_Toc99101381)

[ПРИЛОЖЕНИЕ 3](#_Toc99101382). [РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 78](#_Toc99101383)

[ПРИЛОЖЕНИЕ 4](#_Toc99101384). [ПРОТОКОЛ ИСПЫТАНИЙ ПРОГРАММНОГО СРЕДСТВА 87](#_Toc99101385)

# ВВЕДЕНИЕ

Информационная система (ИС) — [система](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0), предназначенная для хранения, поиска и обработки [информации](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D0%B8%D1%8F), и соответствующие организационные ресурсы (человеческие, технические, финансовые и т. д.), которые обеспечивают и распространяют информацию.

В современном мире информационные системы используются повсеместно, они обеспечивают удобство получения информации и работы с ней. Для каждой области существует своя информационная система учитывающая все особенности сферы в которой она задействована. Информационная система для данной темы будет избыточна, поэтому для данной темы будет проектироваться информационное приложение с минимальным необходимым набором функций.

Основными задачами при создании информационного приложения являются: создание структурированной и систематизированной базы данных; учесть все варианты при приеме заказа сотрудниками салона; организация простого поиска по каталогу, а также выдача отчетов и возможность работы с отчетной информацией.

Постепенно данные накапливаются, и архивное хранение и стандартный поиск вручную информации и необходимой документации является трудоемким и не всегда эффективным способом. Поэтому необходимо разработать приложение, которое позволит сократить время, затрачиваемое на ведение, обработку и поиск данных, формирование необходимой документации и, как следствие, высвобождение этого времени на выполнение других заданий, следовательно, достигая этим повышения эффективности работы.

С экономической точки зрения необходимо отметить, что подобное нововведение позволит значительно облегчить процесс обслуживания клиентов и резко сократит время оказания услуг, что позволяет обслуживать большее количество клиентов за одно и тоже время, и как результат ведет к увеличению прибыли салона.

Отчет по продажам можно просматривать за любой период, тем самым, выявляя товары, пользующиеся наибольшим и наименьшим спросом, и т. д. Дополнительно появляется возможность контроля над сотрудниками и просмотра заказов, осуществленных каждым клиентом.

Разработка информационного приложения будет осуществляться на языке программирования C#, т.к. он предоставляет широкий спектр функциональных возможностей, в том числе работа с Microsoft Excel и Word. Этот язык программирования позволяет не выходя из приложения использовать весь функционал вышеупомянутых программ, что упрощает как работу с данными, так и разработку. Так же данный язык позволяет работать с базами данных разработанными в такой среде как Microsoft Access или Microsoft SQL Server Management Studio.

C# - объектно-ориентированный язык программирования. Разработан в 1998-2001 годах группой инженеров компании Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота как язык разработки приложений для платформы .NET Framework и .NET Core. Впоследствии был стандартизирован как ECMA-334 и ISO/IEC 23270.

C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, переменные, свойства, обобщенные типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML.

SQL Server Management Studio (SSMS) — утилита из [Microsoft SQL Server](https://ru.wikipedia.org/wiki/Microsoft_SQL_Server) 2005 и более поздних версий для конфигурирования, управления и [администрирования](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%BD%D0%BE%D0%B5_%D0%B0%D0%B4%D0%BC%D0%B8%D0%BD%D0%B8%D1%81%D1%82%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) всех компонентов Microsoft SQL Server. Утилита включает [скриптовый](https://ru.wikipedia.org/wiki/%D0%A1%D0%BA%D1%80%D0%B8%D0%BF%D1%82) редактор и графическую программу, которая работает с объектами и настройками сервера.

Главным инструментом SQL Server Management Studio является Object Explorer, который позволяет пользователю просматривать, извлекать объекты сервера, а также полностью ими управлять.

Также есть SQL Server Management Studio Express для Express версии сервера, которая является бесплатной. Однако в ней нет поддержки ряда компонентов (Analysis Services, Integration Services, Notification Services, Reporting Services) и SQL Server 2005 Mobile Edition.

Microsoft SQL Server – система управления реляционными базами данных (РСУБД), разработанная корпорацией Microsoft. Основной используемый язык запросов – Transact-SQL, создан совместно с Microsoft и Sybase. Transact-SQL является реализацией стандарта ANSI/ISO по структурированному языку запросов (SQL) с расширениями. Используется для работы с базами данных от персональных до крупных баз данных масштаба предприятия, конкурирует с другими СУБД в этом сегменте рынка.

SQL – декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных.

Является, прежде всего, информационно-логическим языком, предназначенным для описания, изменения и извлечения данных, хранимых в реляционных базах данных. В общем случае SQL (без ряда современных расширений) считается языком программирования на полным по Тьюрингу, но вместе с тем стандарт языка спецификацией SQL/PSM предусматривает возможность его процедурных расширений.

Изначально SQL был основным способом работы пользователя с [базой данных](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) и позволял выполнять следующий набор операций:

* создание в базе данных новой таблицы;
* добавление в таблицу новых записей;
* изменение записей;
* удаление записей;
* выборка записей из одной или нескольких таблиц (в соответствии с заданным условием);
* изменение структур таблиц.

Со временем SQL усложнился - обогатился новыми конструкциями, обеспечил возможность описания и управления новыми хранимыми объектами (например, индексы, представления, триггеры и хранимые процедуры) — и стал приобретать черты, свойственные языкам программирования.

При всех своих изменениях SQL остаётся самым распространённым лингвистическим средством для взаимодействия прикладного программного обеспечения с базами данных. В то же время современные [СУБД](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D0%B1%D0%B0%D0%B7%D0%B0%D0%BC%D0%B8_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85), а также информационные системы, использующие СУБД, предоставляют пользователю развитые средства визуального построения запросов.

Целью дипломного проекта является внедрение информационных технологий в процесс работы автомобильного салона, а также анализ его деятельности и оптимизация работы с данными и клиентами. Результатом выполнения работы является создание готовой к эксплуатации информационного приложения для автолюбителей.

Для достижения цели необходимо решить следующие задачи:

- изучить предметную область;

- создать функциональную модель;

- разработать информационное приложение с помощью языка программирования C#;

- провести тестирование и отладку приложения.

# ИССЛЕДОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

Разрабатываемое информационное приложение предназначено для салонов (дилеров) автомобилей занимающихся продажей автомобилей.

Изучив несколько информационных приложений разных производителей автомобилей (Porsche, Geely, Nissan) можно выявить несколько схожих признаков. У всех приложений выдавались разные технические характеристики автомобилей, тоже самое и с фильтром, за исключением некоторых технических параметров, таких как: модель, двигатель, кузовная модель, мощность, разгон, расход топлива, тип коробки передач, сторона рулевого механизма, тип привода, количество мест, год производства и стоимость. Это и будет минимальный необходимый набор технических характеристик и фильтров. Так же в каждом приложении можно было оформить покупку единовременным платежом или взять автомобиль в лизинг, что и будет реализовано в информационном приложении.

Главная функция информационного приложения – продажа автомобилей и предоставление информации о каталоге продаваемых салоном автомобилей. Она предполагает предоставление полной информации о продаваемом автомобиле, учет заказов, учет сотрудников, учет поставщиков, учет автопарка салона, учет идентификационных номеров автомобилей, ведение документов и формирование отчетов.

Эта функция разделяется на несколько подпунктов:

1. Каталог автомобилей салона – функция предполагает перечень продаваемых автомобилей и предоставление всей информации об их стоимости и характеристиках (Номер модели, марка и т.д.);
2. Учет сотрудников салона – функция предполагает общую информацию и личные логины и пароли сотрудников (Идентификационный номер, ФИО и т.д.);
3. Учет клиентов – функция предполагает ввод данных общей информации о клиентах (Паспортные данные, ФИО и т.д.);
4. Учет заказов – функция предполагает перечень заказов и их текущий статус (Номер, дата, VIN автомобиля статус, клиент и т.д.);
5. Учет поставщиков – функция предполагает общую информацию о поставщиках (Наименование, адрес и т.д.);
6. Учет идентификационных номеров автомобилей – функция предполагает перечень уникальных номеров автомобилей (VIN, номер модели);
7. Ведение документов – функция предполагает оформление и вывод на печать документов, таких как характеристики автомобиля, акты о продаже автомобиля, документы о долговых обязательствах и т.д.;
8. Формирование отчетов – функция предполагает быстрое формирование отчетов по продажам и анализ данных на заданный период.

# ПОСТАНОВКА ЗАДАЧИ И ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

## Техническое задание

* + 1. Цель создания дипломного проекта

Главной целью дипломного проекта является создание «Информационного приложения для автолюбителей», которое облегчит работу сотрудникам разных автомобильных салонов и поможет покупателям определиться в выборе транспортного средства.

* + 1. Назначение и область применения

Наименование приложения – «Информационное приложение для автолюбителей». Данное приложение предназначено для отображения информации об автомобилях салона: о каталоге салона; для учета клиентов; для учета поставщиков; для учета заказов; для учета сотрудников; для предоставления отчета по продажам и т.д.

Пользователями программного продукта являются сотрудники салона, в том числе директор, продавцы-консультанты.

* + 1. Основание для разработки

Тема предложена преподавателем для выполнения дипломного проекта специальности «Программное обеспечение информационных технологий».

## Требования к программе

* + 1. Требования к функциональным характеристикам

Разрабатываемое приложение должно выполнять следующие функции: заполнение каталога автомобилей, списка заказов, поставщиков, клиентов, автопарка салона; аутентификация пользователя по его логину и паролю; заполнение отчетов по продажам и их анализ.

* + 1. Требования к составу и параметрам технических средств

Система должна работать на IBM-совместимых персональных компьютерах.

Требования к техническим характеристикам ПК пользователя:

Минимальная конфигурация:

- тип процессора: Pentium III и выше;

- обьем ОЗУ: 512 Мб и более;

-обьем свободного места на жестком диске: 300 Мб.

Рекомендуемая конфигурация:

˗ тип процессора: Pentium IV и выше;

˗ объём ОЗУ: 1024 Мб и более;

˗ объём свободного места на жёстком диске: 300 Мб.

* + 1. Требования к информационной и программной совместимости

Программа должна работать под управлением семейства операционных систем Win 32/64 (Windows XP/7 и т.п.).

## Требования к программной документации

Состав программной документации должен включать в себя:

- техническое задание;

- программу и методики испытаний;

- руководство оператора.

## Стадии и этапы разработки

На стадии разработки дипломного проекта должен быть выполнены этапы разработки, согласования и утверждения настоящего технического задания. На стадии рабочего проектирования должны быть выполнены перечисленные ниже этапы работы:

1. Разработка программы;
2. Разработка программной документации;
3. Испытания программы.

## Проектирование UML

Принцип взаимодействия сотрудника с программой отражается в следующих диаграммах на рисунках 2.1 – 2.3.



Рисунок 2.1 – Взаимодействие сотрудника и программы

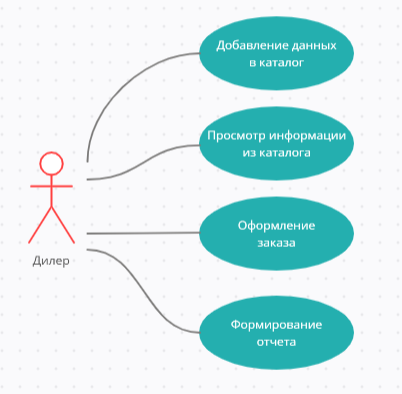


Рисунок 2.2 – Диаграмма вариантов использования

Для того, чтобы реализовать программу следует написать базу данных, в которой будет записан весь каталог автомобилей с характеристиками и модификациями. Также в программе надо реализовать поиск и фильтры по разным параметрам.

## Функциональная модель

Функциональная модель предназначена для изучения особенностей работы (функционирования) системы и её назначения во взаимосвязи с внутренними и внешними элементами.

Функциональная модель показывает, каким образом выходные данные вычисляются по входным данным, не рассматривая порядок и способ реализации вычислений. Она состоит из набора диаграмм потока данных, которые показывают потоки значений от внешних входов через операции и внутренние хранилища данных к внешним выходам.

На основе нотации IDEF0 была разработана контекстная диаграмма, которая показывает входные и выходные ресурсы, правила управления и механизм управления.

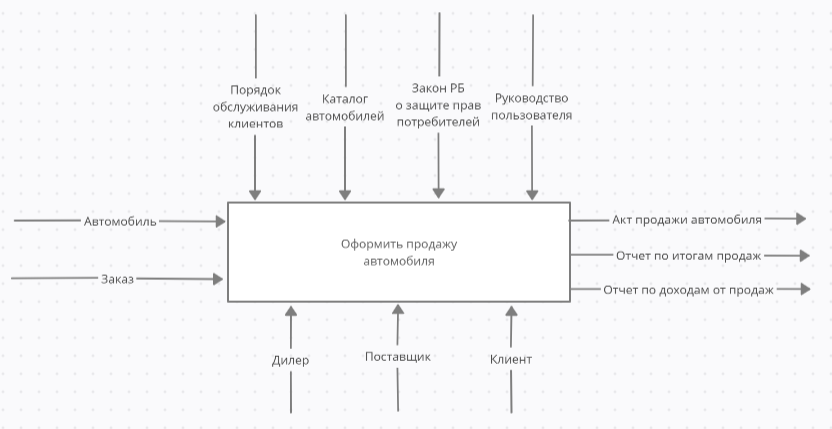


Рисунок 2.3 – Диаграмма IDEF-0

Далее подробно рассмотрим процессы, происходящие внутри салона автомобилей.

Главной задачей салона является оформление заказа автомобиля. Сотрудники салона заносят данные о моделях автомобилей в каталог. Автомобиль поступает в салон от поставщика по заказу клиента, затем сотрудниками салона оформляется в приход и доставляется покупателю.

Создадим диаграмму первого уровня в методологии IDEF.

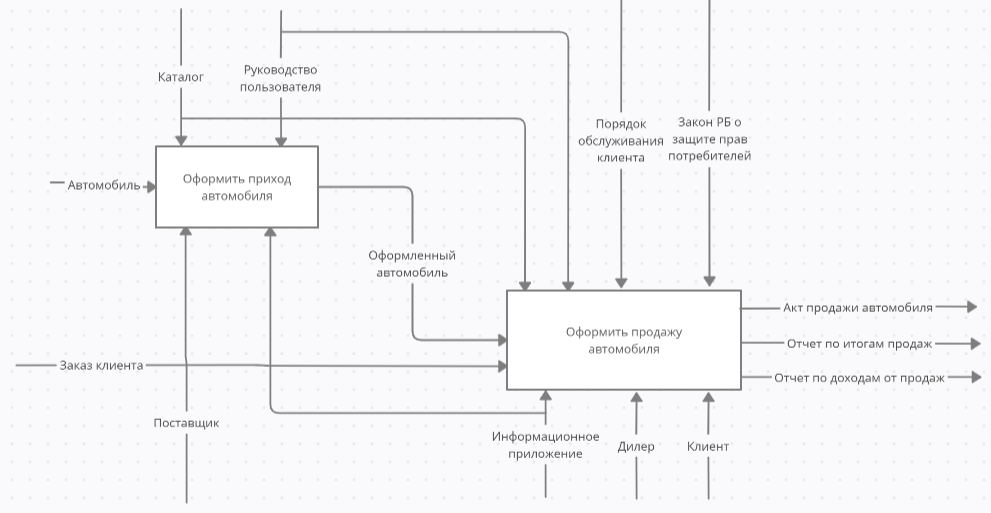


Рисунок 2.4 – Диаграмма IDEF-1

Для более детальной характеристики процесса «Оформить продажу автомобиля» создадим диаграмму второго уровня.

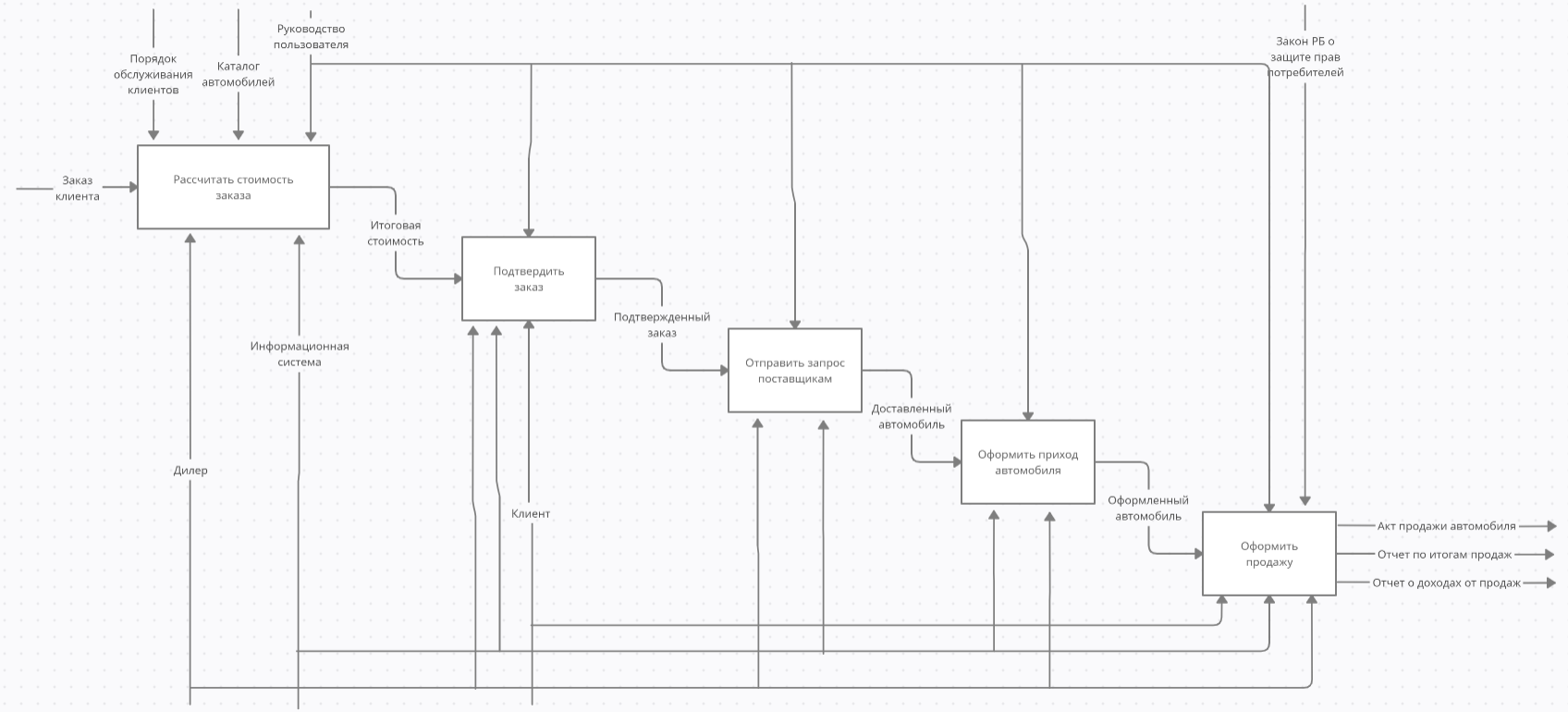


Рисунок 2.5 – Диаграмма IDEF-2 «Оформить продажу автомобиля»

Данная диаграмма отображает все процессы, происходящие в салоне при оформлении продажи автомобиля.

Сотрудник салона рассчитывает стоимость заказа и с подтверждения клиента оформляет заказ как подтвержденный. Затем сотрудник отправляет запрос поставщику на поставку автомобиля. После прихода автомобиля и его оформления, сотрудник связывается с клиентом для оформления документа о покупке и передачи автомобиля. Все данные заносятся в информационную систему.

После оформления продажи автомобиля можно сформировать отчет по итогам продаж и доходах от них.

Также для более детальной характеристики процесса «Оформить приход автомобиля» создадим диаграмму второго уровня.

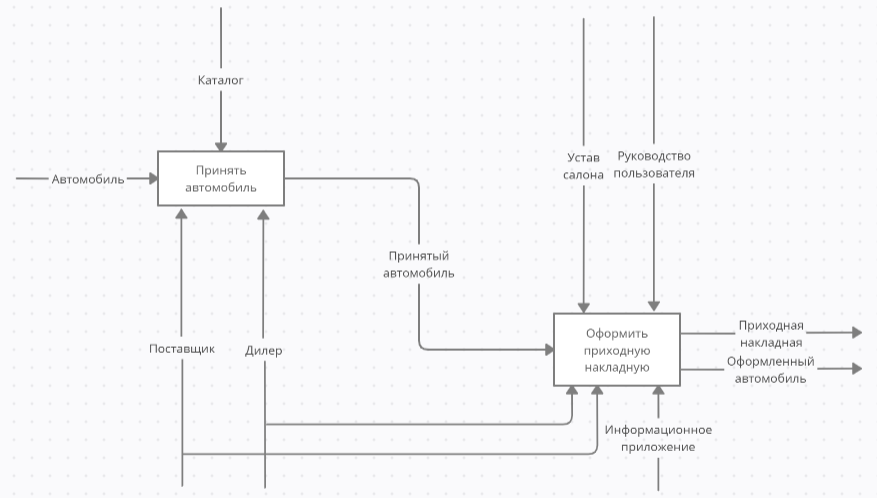


Рисунок 2.6 – Диаграмма IDEF-2 «Оформить приход автомобиля»

Данная диаграмма отображает все процессы, происходящие в салоне при оформлении прихода автомобиля.

Сотрудник салона принимает автомобиль от поставщика, далее по уставу салона на автомобиль оформляется приходная накладная. После этого сотрудник отвозит оформленный автомобиль покупателю.

## Структура базы данных

Структура базы данных (рис. 2.7) необходима для визуального представления информации о будущем справочнике в целом. Для дальнейшей разработки справочника автомобилей следует написать базу данных исходя из рассмотренной предметной области.

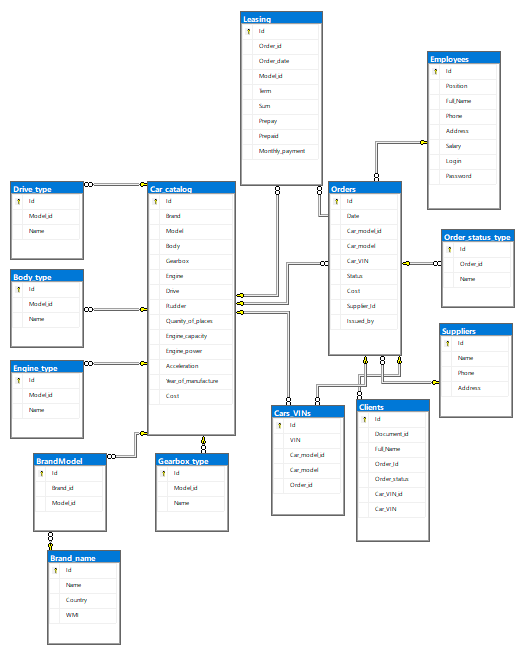


Рисунок 2.7 – Структура базы данных

1. РЕАЛИЗАЦИЯ. ОПИСАНИЕ АЛГОРИТМОВ РАБОТЫ ПРОГРАММНОГО СРЕДСТВА

## Алгоритм работы ПО

Алгоритм - набор инструкций, описывающих порядок действий исполнителя для достижения результата решения задачи за конечное число действий. В старой трактовке вместо слова «порядок» использовалось слово «последовательность», но по мере развития параллельности в работе компьютеров слово «последовательность» стали заменять более общим словом «порядок». Это связано с тем, что работа каких-то инструкций алгоритма может быть зависима от других инструкций или результатов их работы.

Таким образом, некоторые инструкции должны выполняться строго после завершения работы инструкций, от которых они зависят. Независимые инструкции или инструкции, ставшие независимыми из-за завершения работы инструкций, от которых они зависят, могут выполняться в произвольном порядке, параллельно или одновременно, если это позволяют используемые процессор и операционная система.

Алгоритм означает точное описание некоторого процесса, инструкцию по его выполнению. Разработка алгоритма является сложным и трудоемким процессом. Алгоритмизация - это техника разработки (составления) алгоритма для решения задач на ЭВМ.

Алгоритм работы данного ПО состоит в следующем: сотрудник с помощью поиска или фильтров находит определенную модель автомобиля. Кликнув на картинку автомобиля, в программе разворачиваются характеристики автомобиля и его модификации.

## Общие сведения

- Обозначение программы: Chess.exe

- Наименование программы: Chess

- Автор: Побединский А.В.

- Дата создания: 05.07.2021г.

- Программная среда: Visual Studio 2019.

- Программное обеспечение: Windows.

- Язык программирования: C#.

# Функциональное назначение

Программа для поиска автомобилей и информирование о их характеристиках и модификациях.

## Требование к языку в среде разработки

Программа была написана на языке программирования “C#”, среда кроссплатформенной разработки Visual Studio.

Visual Studio — это стартовая площадка для написания, отладки и сборки кода, а также последующей публикации приложений. Интегрированная среда разработки (IDE) представляет собой многофункциональную программу, которую можно использовать для различных аспектов разработки программного обеспечения. Помимо стандартного редактора и отладчика, которые существуют в большинстве сред IDE, Visual Studio включает в себя компиляторы, средства автозавершения кода, графические конструкторы и многие другие функции для упрощения процесса разработки.

Функциональная структура среды включает в себя:

Редактор исходного кода, который включает множество дополнительных функций, как автодополнение IntelliSense, рефакторинг кода и т. д.;

* отладчик кода;
* редактор форм, предназначенный для упрощённого конструирования графических интерфейсов;
* веб-редактор;
* дизайнер классов;
* дизайнер схем баз данных.

Visual Studio также позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода (Subversion и Visual Source Safe), добавление новых наборов инструментов (для редактирования и визуального проектирования кода на предметно-ориентированных языках программирования или инструментов для прочих аспектов процесса разработки программного обеспечения). Интегрированная среда разработки (Integrated Development Environment - IDE) Visual Studio предлагает ряд высокоуровневых функциональных возможностей, которые выходят за рамки базового управления кодом.

Ниже перечислены основные преимущества IDE-среды Visual Studio:

- Встроенный Web-сервер. Для обслуживания Web-приложения ASP.NET необходим Web-сервер, который будет ожидать Web-запросы и обрабатывать соответствующие страницы. Наличие в Visual Studio интегрированного Web-сервера позволяет запускать Web-сайт прямо из среды проектирования, а также повышает безопасность, исключая вероятность получения доступа к тестовомуWeb-сайту с какого-нибудь внешнего компьютера, поскольку тестовый сервер может принимать соединения только с локального компьютера. Поддержка множества языков при разработке. Visual Studio позволяет писать код на своем языке или любых других предпочитаемых языках, используя все время один и тот же интерфейс (IDE). Более того, Visual Studio также еще позволяет создавать Web-страницы на разных языках, но помещать их все в одно и то же Web-приложение. Единственным ограничением является то, что в каждой Web-странице можно использовать только какой-то один язык (очевидно, что в противном случае проблем при компиляции было бы просто не избежать).

- Меньше кода для написания. Для создания большинства приложений требуется приличное количество стандартного стереотипного кода, и Web-страницы ASP. NET тому не исключение. Например, добавление Web-элемента управления, присоединение обработчиков событий и корректировка форматирования требует установки в разметке страницы ряда деталей. В Visual Studio такие детали устанавливаются автоматически.

- Интуитивный стиль кодирования. По умолчанию Visual Studio форматирует код по мере его ввода, автоматически вставляя необходимые отступы и применяя цветовое кодирование для выделения элементов типа комментариев. Такие незначительные отличия делают код более удобным для чтения и менее подверженным ошибкам. Применяемые Visual Studio автоматически параметры форматирования можно даже настраивать, что очень удобно в случаях, когда разработчик предпочитает другой стиль размещения скобок (например, стиль K&R, при котором открывающая скобка размещается на той же строке, что и объявление, которому она предшествует).

- Более высокая скорость разработки. Многие из функциональных возможностей Visual Studio направлены на то, чтобы помогать разработчику делать свою работу как можно быстрее. Удобные функции, вроде функции IntelliSense (которая умеет перехватывать ошибки и предлагать правильные варианты), функции поиска и замены (которая позволяет отыскивать ключевые слова как в одном файле, так и во всем проекте) и функции автоматического добавления и удаления комментариев (которая может временно скрывать блоки кода), позволяют разработчику работать быстро и эффективно.

- Возможности отладки. Предлагаемые в Visual Studio инструменты отладки являются наилучшим средством для отслеживания загадочных ошибок и диагностирования странного поведения. Разработчик может выполнять свой код по строке за раз, устанавливать интеллектуальные точки прерывания, при желании сохраняя их для использования в будущем, и в любое время просматривать текущую информацию из памяти.

Visual Studio также имеет и множество других функций: возможность управления проектом; встроенная функция управления исходным кодом; возможность рефакторизации кода; мощная модель расширяемости. Более того, в случае использования Visual Studio 2008 Team System разработчик получает расширенные возможности для модульного тестирования, совместной работы и управления версиями кода (что значительно больше того, что предлагается в более простых инструментах вроде Visual SourceSafe).

В качестве недостатка можно отметить невозможность отладчика (Microsoft Visual Studio Debugger) отслеживать в коде режима ядра. Отладка в Windows в режиме ядра в общем случае выполняется при использовании WinDbg, KD или SoftICE.

## Описание таблиц базы данных

Car\_catalog – главная таблица, которая будет отображать как описание модели автомобиля (таблица 3.1).

Таблица 3.1 – Car\_catalog

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Длина | Описание |
| Id | Int |  | Ключевое поле |
| Brand | Varchar | 255 | Марка автомобиля |
| Body | Varchar | 255 | Тип кузова |
| Gearbox | Varchar | 255 | Тип коробки передач |
| Engine | Varchar | 255 | Тип двигателя |
| Drive | Varchar | 255 | Привод |
| Rudder | Varchar | 255 | Руль |
| Quanity\_of\_places | Int |  | Количество мест |
| Engine\_capacity | Numeric | 3, 2 | Расход |
| Engine\_power | Numeric | 18 | Мощность |
| Acceleration | Numeric | 5,2 | Разгон |
| Year\_of\_manufacture | Date |  | Год производства |
| Cost | Money |  | Стоимость |

Brand\_name – таблица с названием всех брендов автомобилей (таблица 3.2).

Таблица 3.2 – Brand\_name

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Длина | Описание |
| Id | Int |  | Ключевое поле |
| Name | Varchar | 255 | Название |
| Country | Varchar | 255 | Страна |
| WMI | Varchar | 255 | Код страны |

BrandModel – таблица поддерживающая связь между таблицами Brand\_name и Car\_catalog (таблица 3.3).

Таблица 3.3 – BrandModel

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Длина | Описание |
| Id | Int |  | Ключевое поле |
| Brand\_id | Int |  | Id брэнда |
| Model\_id | Int |  | Id модели |

Body\_type – таблица с типами кузовов автомобилей (таблица 3.4).

Таблица 3.4 – Body\_type

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Длина | Описание |
| Id | Int |  | Ключевое поле |
| Model\_id | Int |  | Id модели |
| Name | Varchar | 255 | Название |

Gearbox\_type – таблица с типами автоматических коробок передач (Auto, Robot, Variable) (таблица 3.5).

Таблица 3.5 – Gearbox\_type

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Длина | Описание |
| Id | Int |  | Ключевое поле |
| Model\_id | Int |  | Id модели |
| Name | Varchar | 255 |  |

Drive\_type – таблица с типами привод автомобилей (Four/Front/Rear-wheel) (таблица 3.6).

Таблица 3.6 – Drive type

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Длина | Описание |
| Id | Int |  | Ключевое поле |
| Model\_id | Int |  | Id модели |
| Name | Varchar | 255 | Название |

Engine\_type – таблица с типами двигателей (таблица 3.7).

Таблица 3.7 – Engine type

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Длина | Описание |
| Id | Int |  | Ключевое поле |
| Model\_id | Int |  | Id модели |
| Name | Varchar | 255 | Название |

Orders – таблица заказов (таблица 3.8).

Таблица 3.8 – Orders

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Длина | Описание |
| Id | Int |  | Ключевое поле |
| Date | Date |  | Дата заказа |
| Car\_model\_id | Int |  | Id модели |
| Car\_model | Varchar | 255 | Название модели |
| Car\_VIN | Varchar | 255 | Идентификационный номер |
| Status | Varchar | 255 | Статус заказа |
| Cost | Money |  | Стоимость |
| Supplier\_id | Int |  | Id поставщика |
| Issued\_by | Int |  | Id сотрудника |

Order\_status\_type – таблица статусов заказов (таблица 3.9).

Таблица 3.9 – Order\_status\_type

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Длина | Описание |
| Id | Int |  | Ключевое поле |
| Order\_id | Int |  | Id заказа |
| Name | Varchar | 255 | Название |

Leasing – таблица заказов оформленных в лизинг (таблица 3.10).

Таблица 3.10 – Leasing

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Длина | Описание |
| Id | Int |  | Ключевое поле |
| Order\_id | Int |  | Id модели |
| Order\_date | Date |  | Дата заказа |
| Model\_id | Int |  | Номер модели |
| Term | Int |  | Срок лизинга |
| Sum | Money |  | Сумма лизинга |
| Prepay |  |  | Сумма предоплаты |
| Prepaid | Tinyint |  | Получена ли предоплата |
| Monthly\_payment |  |  | Ежемесячная плата |

Employees – таблица сотрудников (таблица 3.11).

Таблица 3.11 – Employees

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Длина | Описание |
| Id | Int |  | Ключевое поле |
| Position | Varchar | 255 | Должность |
| Full\_Name | Varchar | 255 | ФИО |
| Phone | Varchar | 255 | Телефонный номер |
| Address | Varchar | 255 | Адрес |
| Salary | Money |  | Зарплата |
| Login | Varchar | 255 | Логин |
| Password | Varchar | 255 | Пароль |

Suppliers – таблица поставщиков (таблица 3.12).

Таблица 3.12 – Suppliers

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Длина | Описание |
| Id | Int |  | Ключевое поле |
| Name | Varchar | 255 | Название |
| Phone | Varchar | 255 | Телефон |
| Address | Varchar | 255 | Адрес |

Clients – таблица клиентов (таблица 3.13).

Таблица 3.13 – Clients

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Длина | Описание |
| Id | Int |  | Ключевое поле |
| Document\_id | Varchar | 255 | Паспортный номер |
| Full\_Name | Varchar | 255 | ФИО |
| Order\_id | Int |  | Id заказа |
| Order\_status | Varchar | 255 | Статус заказа |
| Car\_VIN\_id | Int |  | Id идентификационного номера |
| Car\_VIN | Varchar | 255 | Идентификационный номер |

Cars\_VINs – таблица идентификационных номеров автомобилей (таблица 3.14).

Таблица 3.14 – Cars\_VINs

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Длина | Описание |
| Id | Int |  | Ключевое поле |
| VIN | Varchar | 255 | Идентификационный номер |
| Car\_model\_id | Int |  | Id модели |
| Car\_model | Varchar | 255 | Название модели |
| Order\_id | Int |  | Id заказа |

* 1. Описание связей базы данных

Таблица 3.15 – Связи базы данных

|  |  |  |
| --- | --- | --- |
| Первичный ключ | Вторичный ключ | Тип связи |
| Car\_catalog.Id | BrandModel.Model\_id | Один ко многим |
| BrandModel.Brand\_id | Brand\_name.Id | Один ко многим |
| Car\_catalog.Body | Body\_type.Name | Один ко многих |
| Car\_catalog.Gearbox | Gearbox\_type.Name | Один ко многим |
| Car\_catalog.Engine | Engine\_type.Name | Один ко многим |
| Car\_catalog.Drive | Drive\_type,Name | Один ко многим |
| Car\_catalog.Id | Leasing.Model\_id | Один ко многим |
| Car\_catalog.Id | Orders.Car\_model\_id | Один ко многим |
| Car\_catalog.Id | Cars\_VINs.Car\_model\_id | Один ко многим |
| Orders.Id | Leasing.Order\_id | Один ко многим |
| Orders.Status | Order\_status\_type.Name | Один ко многим |
| Orders.Car\_VIN | Cars\_VINs.VIN | Один ко многим |
| Orders.Id | Clients.Order\_id | Один ко многим |
| Employees.Id | Orders.Issued\_by | Один ко многим |
| Suppliers.Id | Orders.Suppliers\_id | Один ко многим |
| Cars\_VINs.Id | Clients.Car\_VIN\_id | Один ко многим |

## Описание алгоритма работы с приложением

Рассмотрим процесс работы с приложением. В автомобильный салон приходить потенциальный покупатель, сотрудник салона предложит клиенту ознакомиться с ассортиментом автомобилей салона через локальное приложение на стенде. Покупатель, может посмотреть несколько вариантов, и определившись со своим выбором, оформить договор купли-продажи или оформить машину в лизинг. Сотрудник же помогает покупателю оформить договора, оформляет заказ, информирует покупателя о предположительной дате прихода автомобиля и, по приходу автомобиля, доставляет его покупателю на заранее обговоренный адрес.

## Описание основных модулей программы

Модуль – функционально законченный фрагмент программы, оформленный в виде отдельного файла с исходным кодом, предназначенный для использования в других программах.

Рассмотрим несколько основных модулей программы:

- CarInfoForm.cs – главная форма, содержит все дочерние формы

- CarForm.cs – дочерняя форма, отражающая информацию о модели автомобиля.

- CatalogForm.cs – дочерняя форма, содержащая перечень всех моделей автомобилей.

- LoginForm.cs – форма входа в программу, также используется для регистрации новых логинов и паролей.

- TableForm.cs – дочерняя форма, содержит таблицу данных.

- SQLHelper.cs – модуль, содержит логику работы sql запросов.

- Program.cs – главный исполняющий модуль.

## Проектирование интерфейса программного средства

Пользовательский интерфейс (UI) – это интерфейс, с помощью которого человек может управлять программным обеспечением или аппаратным оснащением. Термин в первую очередь означает множество кнопок, окон, вкладок и других вариантов меню, которые контролируются игроком при игре. Кроме того, UI должны быть удобными в использовании, чтобы взаимодействие с ними происходило на максимально интуитивном уровне.

Главной задачей при разработке пользовательского интерфейса было создание простого, но в то же время уникального и понятного внешнего вида графической составляющей приложения.

Логин форма (рис.3.1)– первая форма, открывающаяся перед пользователем. В данной форме пользователь должен ввести логин и пароль собственного аккаунта, в противном случает приложение закроется.

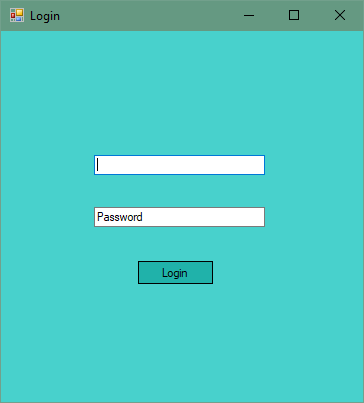


Рисунок 3.1 – Логин форма

При несоответствии полей текст в них окрашивается в красный, при соответствии - в зеленый. На поля не обязательно наводить мышкой, т.к. при окончании ввода и нажатии Enter программа автоматически переходит с первого поля на второе и далее на кнопку входа. При успешном входе программа закрывает форму логин и открывает главную форму.

Главная форма (рис. 3.2) – основная форма, на которой происходят все действия, такие как добавление дочерних форм с таблицами, добавление формы каталога автомобилей и т.д.

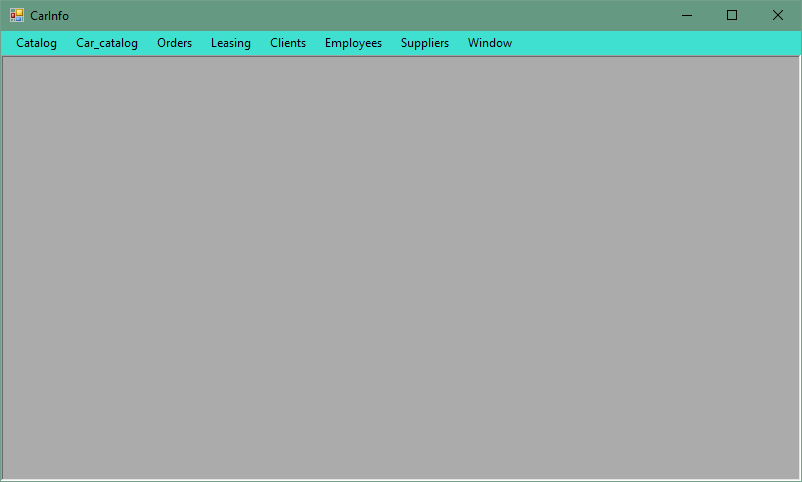


Рисунок 3.2 – Главная форма

В шапке главной формы находится меню вкладок, при нажатии любой кнопки, кроме Window, открывается дочерняя форма, наполнение зависит от названия. Например, при нажатии на кнопку «Catalog» откроется форма с каталогом машин, при нажатии на «Car\_catalog» - таблица со списком моделей и их характеристиками и т.д.

Рассмотрим остальные вкладки по порядку.

Вкладка «Catalog» - содержит каталог автомобилей с их фотографиями и названиями.

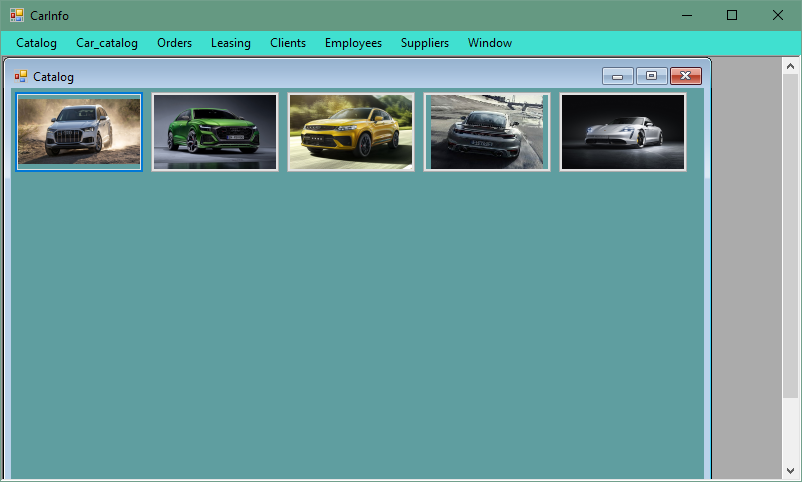


Рисунок 3.3 – Вкладка «Catalog»

При наведении на картинку всплывает надпись с названием марки и модели соответствующего автомобиля. При нажатии открывается форма с характеристиками автомобиля (рисунок 3.4).

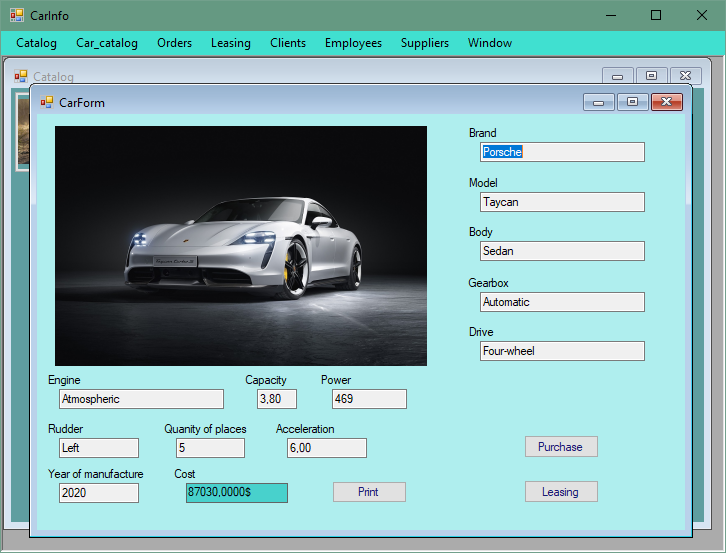


Рисунок 3.4 – Форма с техническими характеристиками

Вкладки «Car\_catalog», «Orders», «Leasing», «Clients», «Employees» и «Suppliers» содержат таблицы с соответствующие их названиям (рис. 3.5).

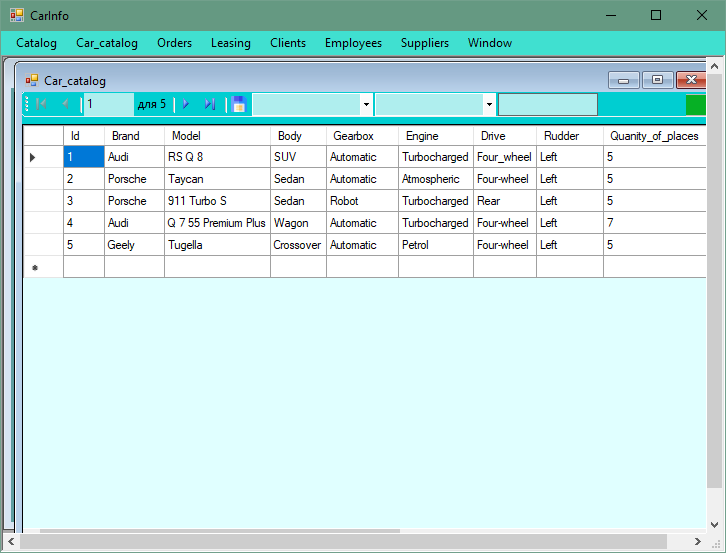


Рисунок 3.5 – Вкладка «Car\_catalog»

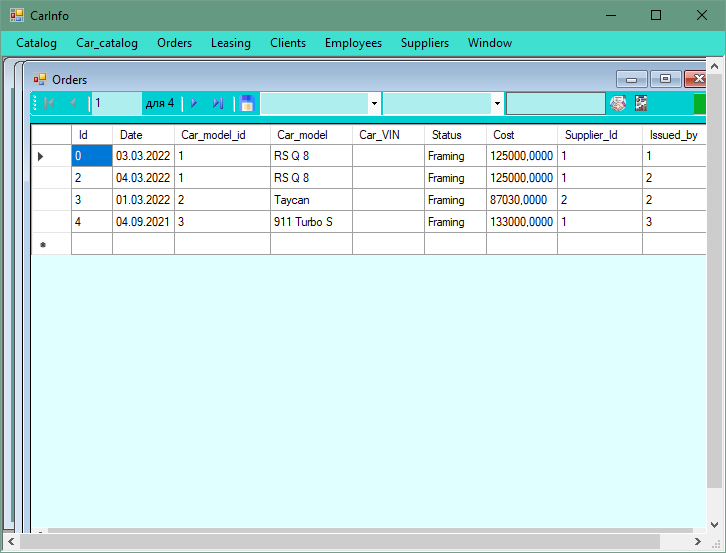


Рисунок 3.6 – Вкладка «Orders»

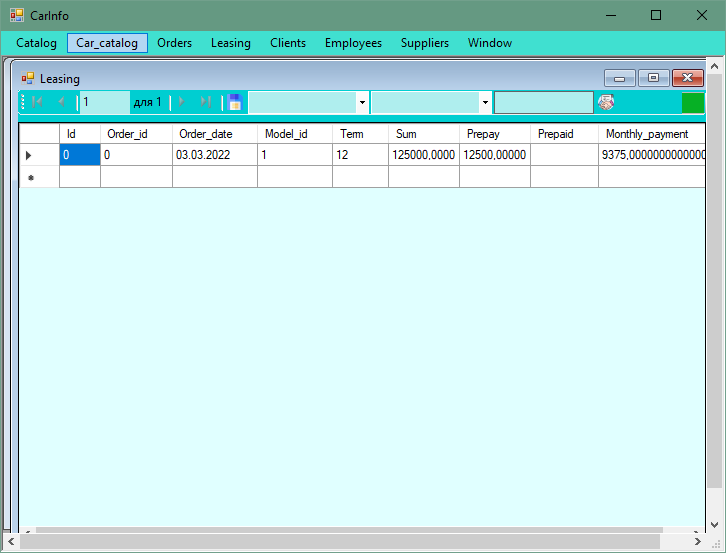


Рисунок 3.7 – Вкладка «Leasing»

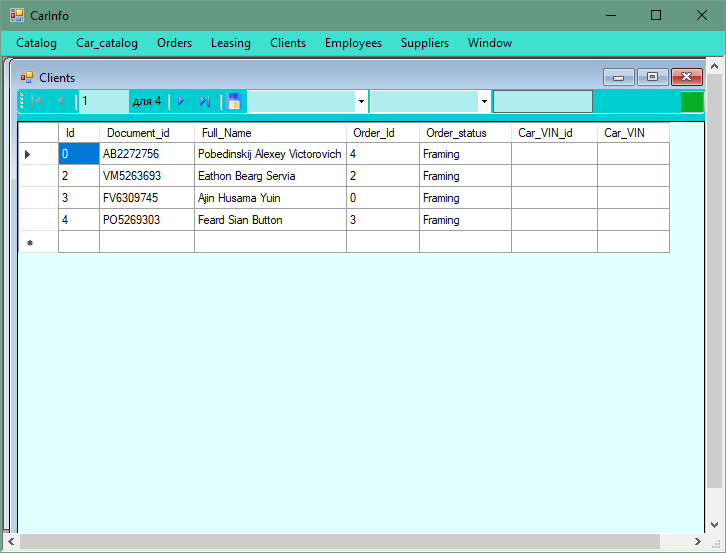


Рисунок 3.8 – Вкладка «Clients»

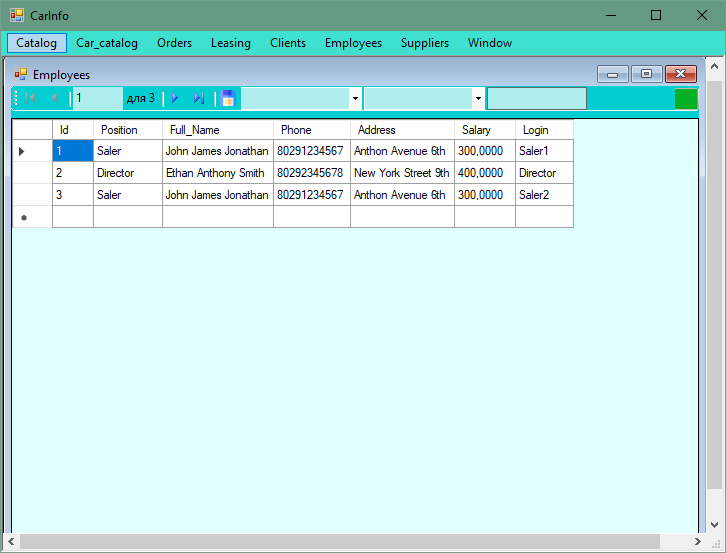


Рисунок 3.9 – Вкладка «Employees»

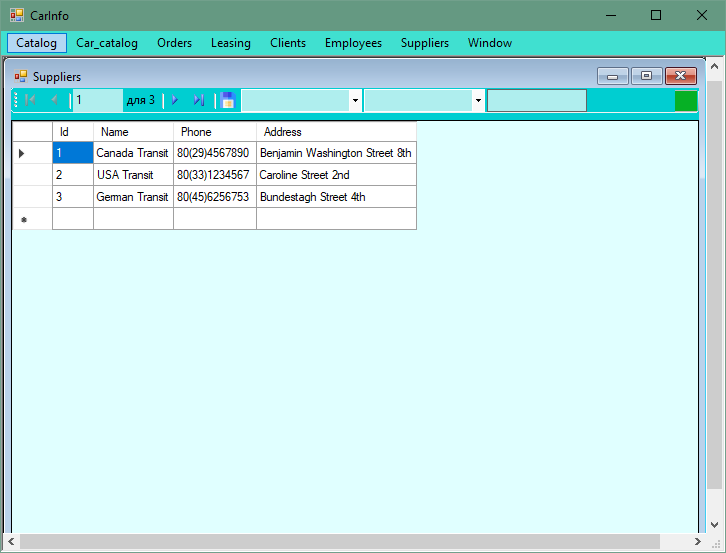


Рисунок 3.10 – Вкладка «Suppliers»

На каждой из таких форм в шапке присутствуют кнопка сохранения данных, 2 поля с выборкой и 1 поле для ввода. При наведении на любой из этих объектов будет всплывать надпись с их названием. Сохранение данных происходит после окончания редактирования записи с помощью кнопки Enter по нажатию на кнопки «Save» или с помощью комбинации клавиш Ctrl+S. Также существует комбинация клавиш для редактирования ячейки – Ctrl+E. Поле с выборкой, находящееся правее кнопки сохранения содержит варианты значений для активной ячейки, однако пользователь может вводить любое значение. Далее идет поле для выбора колонки фильтра, а затем поле динамического поиска.

Отдельно стоит затронуть вкладки «Orders» и «Employees», т.к. на них имеются дополнительные кнопки «Agreement» и «Report». При нажатии кнопки «Agreement» во вкладке «Orders» открывается форма документа Word «Договор купли-продажи транспортного средства» (рис. 3.11), который заполняется вручную, при нажатии этой же кнопки на вкладке «Leasing» - открывается форма документа Word «Заявка лизингополучателя на приобретение предмета лизинга» (рис. 3.12), которая наполовину заполняется автоматически. При нажатии кнопки «Report», она есть только на вкладке «Orders», формируется отчет о доходах за месяц, выбранный из текущей активной строки таблицы «Orders» (рис.3.13).

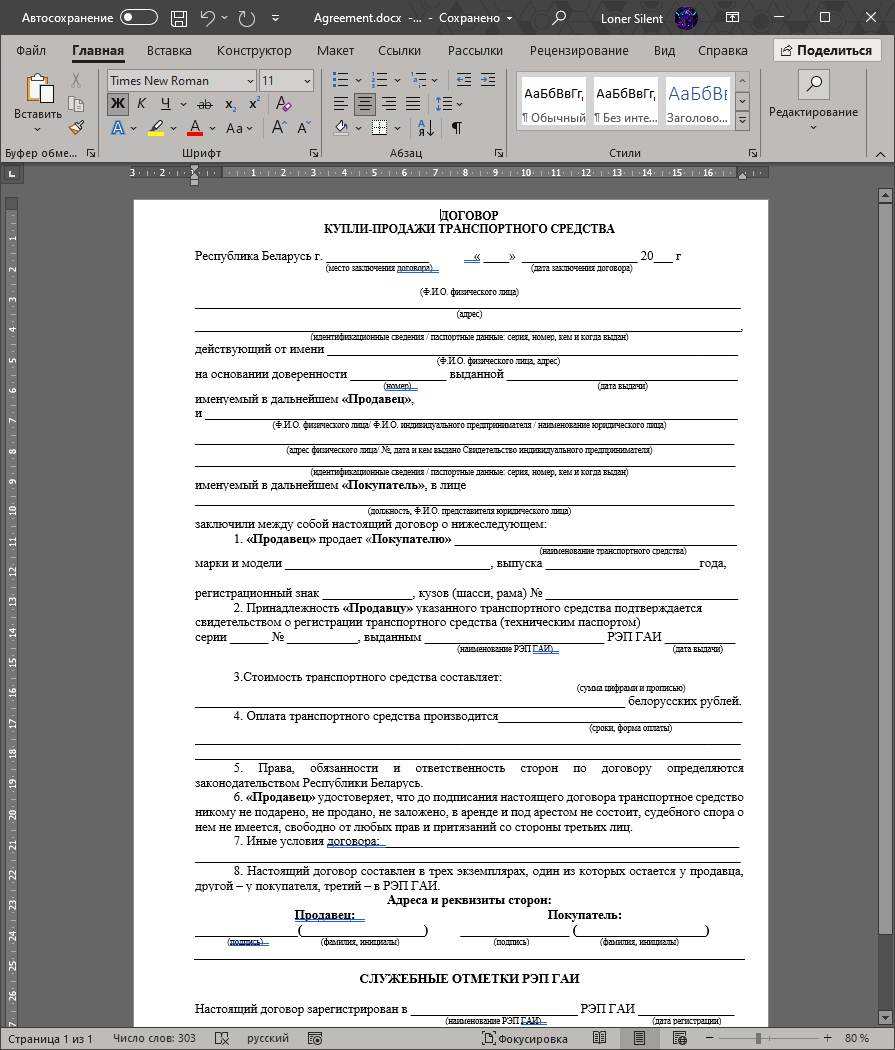


Рисунок 3.11 – Договор купли-продажи

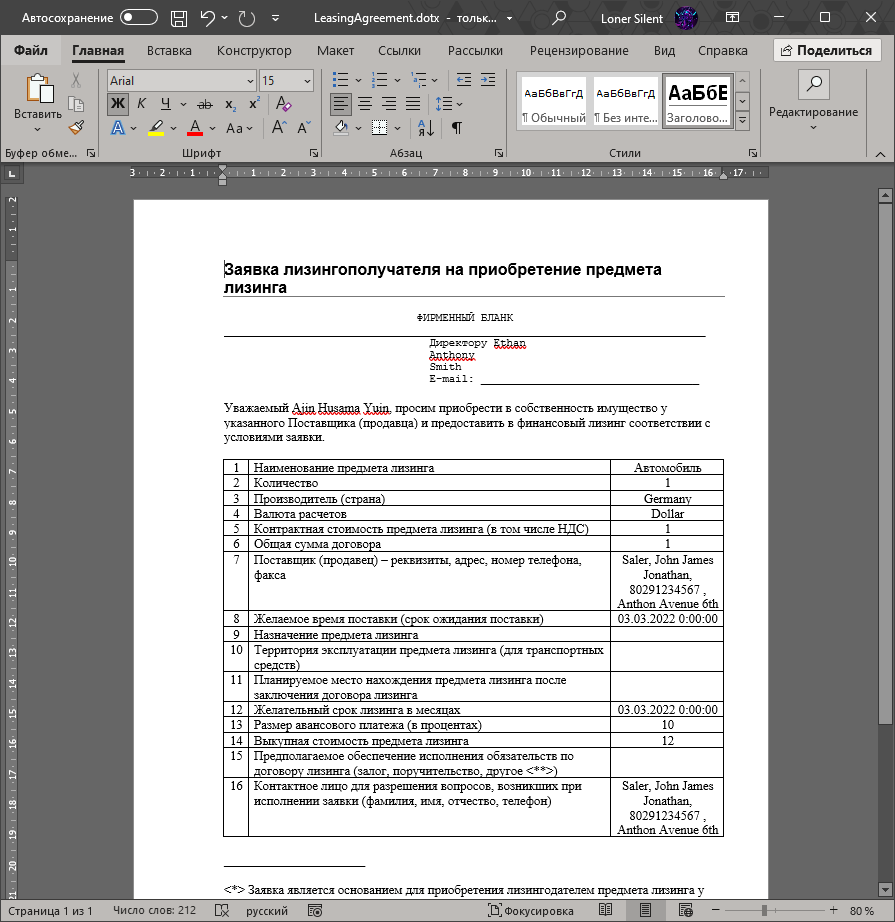


Рисунок 3.12 – Заявка лизингополучателя

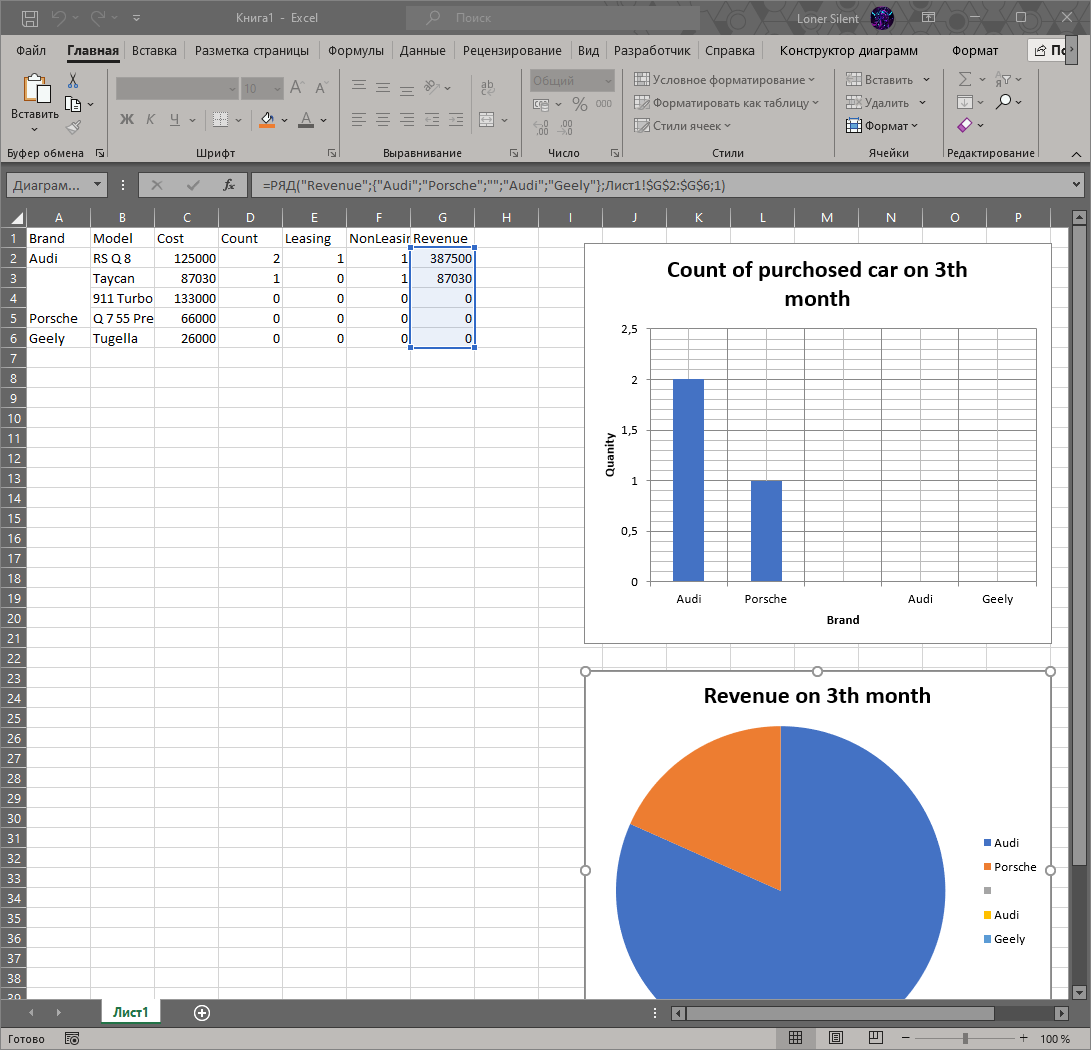


Рисунок 3.13 – «Report»

Таким образом, было разработано информационной приложение «CarInfo» для автолюбителей на языке C# использованием интерфейсов Windows Excel и Word.

Интерфейс приложения выполнен в мягких зеленых тонах, с использованием палитры темных и светлых оттенков цвета Hacky для создания минималистичного и умеренно строгого дизайна. Взаимодействие с приложением интуитивно и понятно.

# ТЕСТИРОВАНИЕ

Тестирование программного обеспечения — процесс проверки соответствия заявленных к продукту требований и реально реализованной функциональности, осуществляемый путем наблюдения за его работой в искусственно созданных ситуациях и на ограниченном наборе тестов, выбранных определенным образом.

В широком смысле, тестирование – это одна из техник контроля качества, которая включает планирование, составление тестов, непосредственно выполнение тестирования и анализ полученных результатов.

Все виды тестирования программного обеспечения, в зависимости от преследуемых целей, можно условно разделить на следующие группы:

1. Функциональные;
2. Нефункциональные;
3. Связанные с изменениями.

Функциональное тестирование – это тестирование ПО, направленное на проверку того, что компонент либо система соответствует функциональным требованиям.

Нефункциональное тестирование – это тестирование атрибутов компонента или системы, не относящихся к функциональности, то есть надежность, эффективность, практичность, сопровождаемость и переносимость, удобство, доступность и т.д.

Тестирование, связанное с изменениями предназначено для проверки исправления дефектов и проверки работоспособности системы после внесения изменений, таких как добавление нового функционала или корректировка старого.

Для нахождения несоответствий в работе программы проведено функциональное тестирование приложения, которое отображено в таблице ниже.

Таблица 4.1 – Результаты тестирования

| Тестируемая функция | Ожидаемый результат | Реальный результат |
| --- | --- | --- |
| 1. Двойной клик ПКМ по ярлыку приложения | Приложение запускается | Приложение запущено |
| 1. Отображение окна входа | Окно входа отображается | Окно входа отображается |
| 1. Возможность ввести логин и пароль | Пользователь может ввести логин и пароль | Пользователь может ввести логин и пароль |
| 1. Выход из приложения при закрытии окна входа | Инициализируется выход и з приложения | Инициализируется выход и з приложения |
| 1. Отображение главного окна приложения | Главное окно отображается | Главное окно отображается |
| 1. Возможность открытия вкладок | Пользователь может открывать вкладки | Пользователь может открывать вкладки |
| 1. Отображение каталога автомобилей | Каталог отображается | Каталог отображается |

Продолжение таблицы 4.1

|  |  |  |
| --- | --- | --- |
| Тестируемая функция | Ожидаемый результат | Реальный результат |
| 1. Возможность открытия окна характеристик автомобиля при нажатии на картинку автомобиля в каталоге | Пользователь может открыть окно характеристик автомобиля при нажатии на картинку автомобиля в каталоге | Пользователь может открыть окно характеристик автомобиля при нажатии на картинку автомобиля в каталоге |
| 1. Отображение характеристик автомобиля | Характеристики отображаются | Характеристики отображаются |
| 1. Возможность распечатать характеристики автомобиля | Документ с характеристиками автомобиля формируется | Документ с характеристиками автомобиля формируется |
| 1. Возможность нажатия на кнопки покупки и оформления лизинга | Кнопки доступны и нажимаются | Кнопки доступны и нажимаются |
| 1. Отображение окон таблиц заказов и лизинга | Таблицы заказов и лизинга отображаются | Таблицы заказов и лизинга отображаются |
| 1. Открытие вкладок с таблицами автомобилей | Вкладки с таблицами открываются | Вкладки с таблицами открываются |
| 1. Возможность редактирования данных таблиц | Пользователь может редактировать данные таблиц | Пользователь может редактировать данные таблиц |
| 1. Возможность нажатия на кнопку сохранения данных таблиц | Пользователь может нажать на кнопку сохранения данных таблиц | Пользователь может нажать на кнопку сохранения данных таблиц |
| 1. Возможность сохранения данных таблиц комбинацией кнопок Ctrl+S | Пользователь может сохранять данные таблиц комбинацией кнопок Ctrl+S | Пользователь может сохранять данные таблиц комбинацией кнопок Ctrl+S |
| 1. Возможность активации клетки для редактирования комбинацией кнопок Ctrl+E | Пользователь может активировать клетку для редактирования комбинацией кнопок Ctrl+E | Пользователь может активировать клетку для редактирования комбинацией кнопок Ctrl+E |
| 1. Возможность выбора значения в поле вариантов | Пользователь может выбрать значение в поле вариантов | Пользователь может выбрать значение в поле вариантов |
| 1. Возможность выбора колонки в поле вариантов фильтра | Пользователь может выбрать колонку для фильтра в поле вариантов | Пользователь может выбрать колонку для фильтра в поле вариантов |
| 1. Возможность ввода текста в поле поиска | Пользователь может вводить текст в поле поиска | Пользователь может вводить текст в поле поиска |

Продолжение таблицы 4.1

| Тестируемая функция | Ожидаемый результат | Реальный результат |
| --- | --- | --- |
| 1. Отображение кнопки «Agreement» на вкладках «Orders» и «Leasing» | Кнопки отображаются и доступны для нажатия | Кнопки отображаются и доступны для нажатия |
| 1. Формирование договора купли-продажи | Договор купли-продажи формируется | Договор купли-продажи формируется |
| 1. Формирование заявки лизингополучателя с автоматическим заполнением полей | Заявка формируется, поля заполняются автоматически | Заявка формируется, поля заполняются автоматически |
| 1. Отображение кнопки «Report» на вкладке «Orders» | Кнопка отображается и доступна для нажатия | Кнопка отображается и доступна для нажатия |
| 1. Формирование отчета при нажатии на кнопку «Report» | Отчет формируется с автозаполнением и отображением диаграмм | Отчет формируется с автозаполнением и отображением диаграмм |
| 1. Отображение выпадающего меню «Window» на главном окне | Меню отображается и доступно для нажатия | Меню отображается и доступно для нажатия |
| 1. Возможность группировки окон вкладок | Окна вкладок группируются в соответствии с выбранным вариантом группировки | Окна вкладок группируются в соответствии с выбранным вариантом группировки |
| 1. Инициализация выхода из приложения | Выход из приложения инициализируется | Выход из приложения инициализируется |

Исходя из результатов, полученных при проведении тестирования, можно сделать вывод, что информационной приложение, разработанное в рамках дипломного проекта, соответствует требованиям и поставленным ранее задачам.

# 5. ЭКОНОМИЧЕСКИЙ РАЗДЕЛ

## 5.1 Определение себестоимости программного обеспечения (ПО), как базы для формирования цены

В современных рыночных экономических условиях ПО выступает преимущественно в виде продукции научно-технических организаций, представляющей собой функционально завершенное и имеющее товарный вид ПО, реализуемое покупателям по рыночным отпускным ценам, установленным в результате анализа рыночных условий и переговоров с заказчиком и возможностей тиражирования. Все завершенные разработки ПО являются научно-технической продукцией.

Широкое применение вычислительной техники требует постоянного обновления и совершенствования ПО. Выбор эффективных проектов ПО связан с их экономической оценкой и расчетом экономического эффекта, который может определяться как у разработчика, так и у пользователя.

У разработчика экономический эффект выступает в виде чистой прибыли, остающейся в распоряжении предприятия от реализации ПО, а у пользователя – в виде экономии трудовых, материальных и финансовых ресурсов, получаемой за счет:

- снижения трудоемкости расчетов и алгоритмизации программирования и отладки программ, за счет использования ПО в процессе разработки автоматизированных систем обработки данных;

- сокращения расходов на оплату машинного времени и других ресурсов на отладку программ;

- снижения расходов на материалы;

- ускорение ввода в эксплуатацию новых систем;

- улучшения показателей основной деятельности предприятий в результате использования ПО.

Стоимостная оценка ПО у разработчиков предполагает составление сметы затрат, которая включает следующие статьи:

- Затраты на материалы;

- Стоимость машинного времени;

- Основная заработная плата;

- Дополнительная заработная плата;

- Отчисление в фонд социальной защиты населения;

- Обязательное страхование от несчастных случаев на производстве и профессиональных заболеваний;

- Прочие затраты;

- Расходы на сопровождение и адаптацию.

Таким образом вычислим себестоимость. При этом исходная информация храниться в таблице 5.1

Таблица 5.1 – исходная информация:

|  |  |  |  |
| --- | --- | --- | --- |
| Показатель | Условное обозначение | Единицы измерения | Значение, рублей |
| Стоимость компьютера | К | Рублей | 1500 |
| Мощность, потребляемая компьютером | МК | КВт/час | 0,70 |
| Полезный фонд времени работы компьютера за год | FK | Час | 2050 |
| Стоимость принтера | П | Рублей | 550 |
| Мощность, потребляемая принтером | МП | КВт/час | 0,09 |
| Полезный фонд времени работы принтера за год | FП | Час | 1050 |
| Стоимость 1 кВт/час (с учетом НДС) | С | Рублей | 0,2968 |
| Норма амортизации для оборудования годовая | Наоб | Проценты | 16,6 |
| Норма амортизации для зданий годовая | Назд | Проценты | 1 |
| Часовая тарифная ставка 13 разряда | ЧТС13 | Рублей | 3,49 |

На основании сметы затрат рассчитывается себестоимость ПО. Расчет ведется в следующей последовательности:

## 5.2 Расчет затрат на материалы

По статье «Материалы» отражаются расходы на магнитные носители, бумагу, красящие ленты и другие материалы, необходимые для разработки ПО. Расчет затрат на материалы (М), необходимые для разработки программного обеспечения (ПО) осуществляется по нормативу (Нр) на 100 команд (принимаем равным 0,482 руб.), с учетом общего объема команд (VкПО ) (берется учащимся по факту, исходя из полученного им количества команд при разработке ПО).

Определение затрат на материалы осуществляется по формуле, в рублях:

М= (1)

## 5.3 Обоснование трудоемкости разработки ПО.

Обоснование трудоемкости разработки ПО ведется следующим образом: на основании опытно-статистического (суммарного) метода, в целом на все разработку, исходя из того, что ее выполняет один программист 2-ой категории. Данные о трудоемкости разработки, определенные опытно-статистическим путём подтверждаю, трудоемкость – 178 часа.

Общий объём программного обеспечения – количество команд (VkПО) – 1016 команд.

Руководитель дипломного проекта \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

5.4 Расчет стоимости машинного времени.

5.4.1 Определение стоимости потребляемой электроэнергии осуществляется по формуле, в рублях:

Сэл= (МК×FK + МП ×FП) × С × Квр×Кс= (0,70 \* 2050 + 0,09 \* 1050) \* 0,2968 \* 0,8 \* 1,05 =

= 381,32 (2)

где Квр - коэффициент, учитывающий использование повремени (Квр=0,8)

Кс - коэффициент, учитывающий потери в сети (Kc=1,05)

5.4.2 Определение суммы амортизационных отчислений осуществляется по формуле, в рублях:

Сао= (К+П) ×Нао/100=(1500+550)\*16,6/100=340,3(3)

Результаты расчетов сводим в таблицу 2.

Таблица 2 - Определение стоимости электроэнергии и амортизационных отчислений

|  |  |  |  |
| --- | --- | --- | --- |
|  | Условное | Значение, рублей | |
| Показатель | обозначение | На год  (ЗМг) | На 1 машино-час (СМч) |
| Стоимость потребляемой электроэнергии | Сэл | 381,32 | 0,19 |
| Сумма ежегодных амортизационных отчислений | Сао | 340,3 | 0,15 |
| Итого: | ЗМг | 721,62 | 0,34 |

Расчет затрат на 1 машино-час ведется исходя из стоимости машинного времени на год (графа 3 таблицы 1) деленный на полезный фонд времени работы компьютера (FK.) (из таблицы «Исходная информация»)

5.4.3 Определение затрат на 1 машино-час осуществляется по формуле, в рублях:

СМч = ЗМг/FK (4)

СМч = 381,32 / 2050 = 0,18

СМч = 304,3 / 2050 = 0,16

5.4.4 Определение стоимости машинного времени осуществляется по формуле, в рублях:

* + 1. СМвр = Тм × СМч = 124,6 \* 0,34 = 42,36(5)

где Тм - машинное время работы в расчёте на программу, час. (70% от трудоемкости разработки программного обеспечения – п.1.2.)

Смги - стоимость одного машино-часа, руб. (итог графы 4 таблицы 2)

## 5.5 Расчет затрат на оплату труда

Расчёт основной заработной платы ведётся исходя из трудоемкости программного обеспечения.

5.5.1 Определение прямой заработной платы осуществляется по формуле, в рублях:

Зпр = Тi × ЧТС13 = 178 \* 3,49 = 621,22 рублей (6)

где Тi – трудоемкость программного обеспечения (из п. 1.2.), ч.

ЧТС13 – часовая тарифная ставка 13 разряда, руб.

Премия определяется как 30 % от основной заработной платы.

5.5.2 Определение премий осуществляется по формуле, в рублях:

П = Зпр × 30 / 100 = 621,22 \* 30 / 100 = 186,37 (7)

Дополнительная заработная плата (Зд) включает выплаты, предусмотренные законодательством о труде (оплата отпусков, льготных часов, времени выполнения государственных обязанностей и других выплат, не связанных с основной деятельностью исполнителей) и определяется по нормативу в процентах к основной заработной плате.

5.5.3 Определение дополнительной заработной платы осуществляется по формуле, в рублях:

где Зос – основная заработная плата;

ЗД – дополнительная заработная плата на конкретное ПО в руб.

НД – норматив дополнительной заработной платы принимается 10% от основной заработной платы.

Отчисления в фонд социальной защиты населения (3C3) определяются в соответствии с действующими законодательными актами по нормативу в процентном отношении к фонду основной и дополнительной зарплаты (Зд).

5.5.4. Определение отчислений в фонд социальной защиты населения осуществляется по формуле, в рублях:

Зсз = (Зос + Зд) \* Нсз / 100 = (807,59 + 80,76) \* 34 / 100 = 302,04 рублей (9)

где Нсз– норматив отчислений в фонд социальной защиты населения (процент действующий норматив на момент расчета).

Обязательное страхование от несчастных случаев на производстве и профессиональных заболеваний (Зстр) берётся по нормативу, установленному в целом по организации (можно принять Нстр– 0,6 %).

5.5.5 Определение отчислений на обязательное страхование от несчастных случаев на производстве и профессиональных заболеваний осуществляется по формуле, в рублях:

Зстр = (Зос + Зд) \* Нстр =(807,59+80,76) \* 0,6% = 5,33 (10)

## 5.6 Расчет полной себестоимости программного обеспечения

Прочие затраты (Пз) включают расходы, связанные с необходимостью содержания аппарата управления, организационные расходы и расходы на общехозяйственные нужды. Принимаются (Нпз) в процентном отношении к основной заработной плате исполнителей. Норматив устанавливается в целом по организации (Нпз принимаем 99%).

5.6.1. Определение прочих затрат осуществляется по формуле, в рублях:

Полная себестоимость без учета расходов на сопровождение и адаптацию (Сп) рассчитывается как сумма затрат на материалы, стоимости машинного времени, затрат на основную, дополнительную заработную плату, стоимости отчислений в фонд социальной защиты населения и на обязательное страхование от несчастных случаев на производстве и профессиональных заболеваний, прочих затрат.

5.6.2 Определение полной себестоимости осуществляется по формуле, в рублях:

Сп = М + СМвр + Зос + Зд + Зсз + Зстр + Пз = 4,90 + 42,36 + 807,59 + 80,76 + 302,04 + 5,33 + 799,51 = 2042,49 рублей (12)

Кроме того, организация-разработчик осуществляет затраты на сопровождение и адаптацию ПО (РСА), которые определяются по нормативу НРСА (принимаем 4%), от полной себестоимости без учёта расходов на адаптацию (Сп).

5.6.3 Определение затрат на сопровождение и адаптацию ПО осуществляется по формуле, в рублях:

Рса = (Сп \* Нрса) / 100 = (2042,49 \* 4) / 100 = 81,70 рублей (12)

где Сп – полная себестоимость ПО без учёта расходов на сопровождение и адаптацию, руб.

5.6.4 Определение общей суммы расходов на разработку (с затратами на сопровождение и адаптацию) (Спл) осуществляется по формуле, в рублях:

Спл = Сп + Рса = 2042,49 +81,7 = 2124,19 рублей (13)

Результаты всех расчётов сводим в таблицу 3.

Таблица 3 – Определение полной себестоимости программного обеспечения

|  |  |  |
| --- | --- | --- |
| Наименование статей затрат | Сумма, руб. | Обоснование расчета |
| 1. Затраты на материалы | 4,90 | пункт 1.1. |
| 2. Стоимость машинного времени | 42,36 | пункт 1.3. |
| 3. Основная заработная плата | 807,59 | пункт 1.4 |
| 4.Дополнительная заработная плата | 80,76 | 10% от ст.3 (пункт 1.4) |
| 5. Итого, фонд оплаты труда (ФОТ) | 888,55 | (ст.3 + ст.4) (пункт 1.4) |
| 6. Отчисление в фонд социальной защиты населения | 302,04 | 34% от ст.5 (пункт 1.4) |
| 7. Обязательное страхование от несчастных случаев на производстве и профессиональных заболеваний | 5,33 | 0,6 % от ст.5  (пункт 1.4) |
| 8. Итого: отчисления на социальные нужды | 307,37 | (ст. 6 + ст.7) (пункт 1.4) |
| 9. Прочие затраты | 799,51 | 99% от ст.3 (пункт 1.5) |
| 10. Полная себестоимость без учёта расходов на сопровождение и адаптацию | 2042,49 | (ст.1+ст.2+ст.5+ст.8  +ст.9) таблицы 3 (пункт 1.5) |
| 11. Расходы на сопровождение и адаптацию. | 81,7 | 4% от ст.10 таблицы 3 (пункт 1.5) |
| 12. Полная себестоимость с учётом расходов на сопровождение и адаптацию | 2124,19 | (ст.10 + ст.11) таблицы 3 (пункт 1.5) |

# ЗАКЛЮЧЕНИЕ

В процессе работы над дипломным проектом было спроектировано и реализовано информационное приложение для автолюбителей. Данное приложение удовлетворяет всем требованиям, предъявленным в задании, и реализует большинство необходимых функций.

Приложение выполняет следующий перечень требований:

- вывод каталога автомобилей и их характеристик;

- учет заказов, сотрудников, клиентов и автомобилей салона;

- вывод документов и отчетов.

В результате проведенной работы был изучены язык взаимодействия с базами данных SQL, проведена работа с проектировщиком баз данных SQL Server Management, приобретены знания и умения работы с COM интерфейсами, такими как Excel и Word, более углубленное понимание работы DataGridView и взаимодействия с данным объектом.

В ходе реализации проекта были выполнены следующие задачи:

* изучены особенности и состояние компьютерной индустрии на сегодняшний день;
* выбраны жанр, вид и платформа для разрабатываемого проекта;
* проведен анализ аналогов;
* выбрано и изучено средство реализации;
* подготовлены необходимые для приложения материалы (база данных, шаблоны документов);
* реализован готовый проект.

Подводя итоги проделанной работы, можно сказать, что все поставленные цели при разработке дипломного проекта были достигнуты. Полученный программный продукт соответствует заданным ранее требованиям и может служить основой для дальнейшей разработки в том же направлении.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Брайан, Н. Привязка данных в Windows Forms [Электронный ресурс] – Режим доступа: <https://drive.google.com/file/d/1zrV03iW8U6QvQZ-HuW6Cl0o7lb7on3hq/view?usp=sharing> ;
2. Официальная документация по языку программирования C# [Электронный ресурс]. – Режим доступа: <https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/> ;
3. Работа с Excel в C# [Электронный ресурс] – Режим доступа: <http://wladm.narod.ru/C_Sharp/comexcel.html#92> ;
4. Форум программистов и сисадминов Cyberforum [Электронный ресурс]. – Режим доступа: <https://www.cyberforum.ru/> .

# ПРИЛОЖЕНИЕ 1

# ИСХОДНЫЙ ТЕКСТ ПРОГРАММНОГО СРЕДСТВА

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using Word = Microsoft.Office.Interop.Word;

using System.Reflection;

namespace CarInfo.Forms

{

public partial class CarForm : Form

{

public string imgFolder;

Queue <string> imgs = new Queue <string> ();

public CarForm()

{

InitializeComponent();

}

public bool FormExist(string form)

{

bool result = false;

foreach (Form f in Application.OpenForms)

{

result = f.Text == form;

if (result == true) { f.Focus(); return result; }

}

return result;

}

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

{

Font = DefaultFont;

ForeColor = Color.Black;

foreach (string img in Directory.GetFiles(imgFolder))

imgs.Enqueue (img);

pictureBox\_MouseClick(null,null);

string modelName = imgFolder.Remove(0,

Directory.GetParent(imgFolder).FullName.Length + 1);

modelBox.Text = modelName;

using (SqlDataReader reader = SQLHelper.ExecuteReader("\*", "Car\_catalog", "Model",

new SqlParameter("@Model", modelName)))

{

reader.Read();

brandBox.Text = reader.GetString(1);

bodyBox.Text = reader.GetString(3);

gearboxBox.Text = reader.GetString(4);

engineBox.Text = reader.GetString(5);

driveBox.Text = reader.GetString(6);

rudderBox.Text = reader.GetString(7);

placesBox.Text = reader.GetInt32(8).ToString();

capacityBox.Text = reader.GetDecimal(9).ToString();

powerBox.Text = reader.GetDecimal(10).ToString();

accelerationBox.Text = reader.GetDecimal(11).ToString();

manufactureBox.Text = reader.GetDateTime(12).Year.ToString();

costBox.Text = reader.GetDecimal(13).ToString() + "$";

reader.Close();

}

}

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

{

if (pictureBox.Image != null) pictureBox.Image.Dispose();

pictureBox.Image = Image.FromFile(imgs.Peek());

imgs.Enqueue(imgs.Dequeue());

}

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

{

if (!FormExist("Orders"))

{

TableForm t = new TableForm

{

MdiParent = this.MdiParent,

Text = "Orders"

};

t.Show();

}

}

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

{

if (!FormExist("Orders"))

{

TableForm f = new TableForm

{

MdiParent = this.MdiParent,

Text = "Orders"

};

f.Show();

}

if (!FormExist("Leasing"))

{

TableForm f = new TableForm

{

MdiParent = this.MdiParent,

Text = "Leasing"

};

f.Show();

}

}

Word.\_Application app;

Word.\_Document doc = null;

readonly string appPath = System.Windows.Forms.Application.StartupPath;

object miss = Type.Missing;

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

{

object fileName = appPath + "\\Docs\\Characteristics.dotx";

app = new Word.Application();

try

{

doc = app.Documents.Open(ref fileName, ref miss, true, ref miss, ref miss, ref miss,

ref miss, ref miss, ref miss, ref miss, ref miss, true, ref miss, ref miss,

ref miss, ref miss);

}

catch (Exception ex)

{

doc.Close(false, ref miss, ref miss);

app.Quit(ref miss, ref miss, ref miss);

doc = null;

app = null;

throw ex;

}

doc.InlineShapes.AddPicture(Directory.GetFiles(imgFolder).First(), ref miss, ref miss,

doc.Bookmarks["photo"].Range);

doc.Bookmarks["brand"].Range.Text = brandBox.Text;

doc.Bookmarks["model"].Range.Text = modelBox.Text;

doc.Bookmarks["body"].Range.Text = bodyBox.Text;

doc.Bookmarks["drive"].Range.Text = driveBox.Text;

doc.Bookmarks["engine"].Range.Text = engineBox.Text;

doc.Bookmarks["capacity"].Range.Text = capacityBox.Text;

doc.Bookmarks["power"].Range.Text = powerBox.Text;

doc.Bookmarks["gearbox"].Range.Text = gearboxBox.Text;

doc.Bookmarks["year"].Range.Text = manufactureBox.Text;

doc.Bookmarks["rudder"].Range.Text = rudderBox.Text;

doc.Bookmarks["places"].Range.Text = placesBox.Text;

doc.Bookmarks["acceleration"].Range.Text = accelerationBox.Text;

doc.Bookmarks["cost"].Range.Text = costBox.Text;

app.Visible = true;

}

}

}

using CarInfo.Forms;

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 CarInfo

{

public partial class CarInfoForm : Form

{

public CarInfoForm()

{

InitializeComponent();

}

public bool formExist(string form)

{

bool result = false;

foreach (Form f in Application.OpenForms)

{

result = f.Text == form ? true : false;

if (result == true) { f.Focus(); return result; }

}

return result;

}

public void SaveAll()

{

foreach (Form f in Application.OpenForms)

{

if (f.GetType().ToString() == "CarInfoForm.TableForm")

{

TableForm q = new TableForm();

q = f as TableForm;

q.bindingNavigatorSaveItem\_Click(null, null);

}

}

}

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

{

if (Info.Position != "Director") employeesToolStripMenuItem.Visible = false;

Font = DefaultFont;

}

private void menuStrip1\_ItemClicked(object sender, ToolStripItemClickedEventArgs e)

{

if (e.ClickedItem.Text == "Catalog")

if (!formExist(e.ClickedItem.Text))

{

CatalogForm f = new CatalogForm();

f.MdiParent = this;

f.Text = e.ClickedItem.Text;

f.Show();

}

if ((e.ClickedItem.Name != "") && (e.ClickedItem.Text != "Window") && (e.ClickedItem.Text != "Catalog"))

if (!formExist(e.ClickedItem.Text))

{

TableForm f = new TableForm();

f.MdiParent = this;

f.Text = e.ClickedItem.Text;

f.Show();

}

}

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

{

this.LayoutMdi(MdiLayout.Cascade);

}

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

{

this.LayoutMdi(MdiLayout.TileHorizontal);

}

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

{

this.LayoutMdi(MdiLayout.TileVertical);

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace CarInfo.Forms

{

public partial class CatalogForm : Form

{

ToolTip tool = new ToolTip();

public CatalogForm()

{

InitializeComponent();

}

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

{

Font = DefaultFont;

string resPath = Path.GetFullPath("Resources");

foreach (string folder in Directory.GetDirectories(resPath))

{

string brand = folder.Remove(0, resPath.Length + 1);

string brandPath = resPath + "\\" + brand;

foreach (string infolder in Directory.GetDirectories(brandPath))

{

Button btn = new Button();

btn.Width = 130;

btn.Height = 82;

btn.Name = infolder;

tool.SetToolTip(btn, brand + " " + infolder.Remove(0, brandPath.Length + 1));

try {

string imgPath = Directory.GetFiles(infolder).First();

btn.BackgroundImage = Image.FromFile(imgPath);

btn.BackgroundImageLayout = ImageLayout.Zoom;

btn.Click += carShow;

} catch (IndexOutOfRangeException) { MessageBox.Show(btn.Text + " has no pictures."); }

catch (InvalidOperationException) { continue; }

flowLayoutPanel1.Controls.Add(btn);

}

}

}

private void carShow(object sender, EventArgs e)

{

if (sender is Button btn)

{

CarForm cf = new CarForm

{

MdiParent = this.MdiParent,

imgFolder = btn.Name

};

cf.Show();

}

}

}

}

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 CarInfo

{

public partial class LoginForm : Form

{

public string Pass;

public string LoginCheck(string login)

{

using (SqlDataReader reader = SQLHelper.ExecuteReader("Password,Position",

"Employees", "Login", new SqlParameter("@Login", login)))

{

string res = null;

reader.Read();

try { Info.Position = reader.GetString(1); res = reader.GetString(0); }

catch (InvalidOperationException) { return null; }

reader.Close();

return res;

}

}

public LoginForm()

{

InitializeComponent();

}

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

{

Font = SystemFonts.DefaultFont;

}

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

{

loginTextBox.Clear();

}

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

{

if (this.Text == "Login")

{

if (loginTextBox.Text == "")

{

loginTextBox.Text = "Login";

loginTextBox.ForeColor = SystemColors.WindowText;

}

else

{

Pass = LoginCheck(loginTextBox.Text);

loginTextBox.ForeColor = Pass

!= null ? Color.Green : Color.Red;

}

}

else if (this.Text == "Register")

{

if (loginTextBox.Text == "")

{

loginTextBox.Text = "Login";

loginTextBox.ForeColor = SystemColors.WindowText;

}

else

{

Pass = LoginCheck(loginTextBox.Text);

loginTextBox.ForeColor = Pass

== null ? Color.Green : Color.Red;

}

}

}

private void loginTextBox\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Enter)

{

loginTextBox\_Leave(sender, e);

passwordTextBox.Focus();

}

}

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

{

passwordTextBox.Clear();

passwordTextBox.UseSystemPasswordChar = true;

}

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

{

if (this.Text == "Login")

{

if (passwordTextBox.Text == "")

{

passwordTextBox.Text = "Password";

passwordTextBox.ForeColor = SystemColors.WindowText;

passwordTextBox.UseSystemPasswordChar = false;

}

else

{

passwordTextBox.ForeColor = Pass

== passwordTextBox.Text ? Color.Green : Color.Red;

}

}

else if (this.Text == "Register")

{

if (passwordTextBox.Text == "")

{

passwordTextBox.Text = "Password";

passwordTextBox.ForeColor = SystemColors.WindowText;

passwordTextBox.UseSystemPasswordChar = false;

}

else if ((passwordTextBox.Text.Length >= 8) && (passwordTextBox.Text.Length <= 12))

{

passwordTextBox.ForeColor = Pass

== null ? Color.Green : Color.Red;

}

else

{

passwordTextBox.ForeColor = Color.Red;

ErrorProvider ep = new ErrorProvider();

ep.SetError(passwordTextBox, "Password length can't be less than 8 or more than 12!");

}

}

}

private void passwordTextBox\_KeyDown(object sender, KeyEventArgs e)

{

if ((e.KeyCode == Keys.Enter) && (passwordTextBox.Text != ""))

{

passwordTextBox\_Leave(sender, e);

loginBtn\_Click(sender, e);

}

}

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

{

if (this.Text == "Login")

{

if ((loginTextBox.ForeColor == Color.Green) && (passwordTextBox.ForeColor

== Color.Green))

{

CarInfoForm cif = new CarInfoForm();

cif.Activate();

this.Dispose();

}

}

else if (this.Text == "Register")

{

TableForm parent = new TableForm();

foreach (Form f in Application.OpenForms)

{

if (f.Text == "Employees") { parent = f as TableForm; break; }

}

DataRow newRow = parent.dt.NewRow();

for (int i = 0; i < 6; i++)

{

newRow[i] = parent.dataGridView.CurrentRow.Cells[i].Value;

}

newRow[6] = loginTextBox.Text;

newRow[7] = passwordTextBox.Text;

parent.dt.Rows.Add(newRow);

parent.bindingNavigatorSaveItem\_Click(null, null);

parent.Focus();

this.Dispose();

}

}

private void LoginForm\_FormClosing(object sender, FormClosingEventArgs e)

{

if ((loginTextBox.ForeColor != Color.Green) && (passwordTextBox.ForeColor != Color.Green)) Application.Exit();

}

}

public static class Info

{

public static string Position { get; set; }

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Reflection;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using Word = Microsoft.Office.Interop.Word;

using Excel = Microsoft.Office.Interop.Excel;

using System.Diagnostics;

namespace CarInfo

{

public partial class TableForm : Form

{

public string tableName;

DataSet ds = new DataSet();

public DataTable dt = new DataTable();

BindingSource bs = new BindingSource();

string oldFolderName;

string oldParentFolderName;

public TableForm()

{

InitializeComponent();

}

private void TableForm\_Resize(object sender, EventArgs e)

{

dataGridView.Width = this.Width - 15;

dataGridView.Height = this.Height - 67;

}

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

{

Font = DefaultFont;

ForeColor = DefaultForeColor;

ds = SQLHelper.LoadData(this.Text);

dt = ds.Tables[0];

bs.DataSource = dt;

this.bindingNavigator.BindingSource = bs;

this.dataGridView.DataSource = dt;

this.dataGridView.AutoResizeColumns();

toolStripProgressBar.Value = 100;

this.dataGridView.Columns["Id"].ReadOnly = true;

if ((this.Text != "Orders") && (this.Text != "Leasing")) agreementStripButton.Visible = false;

if (this.Text != "Orders") reportStripButton.Visible = false;

if (this.Text == "Clients") this.dataGridView.Columns["Order\_status"].ReadOnly = true;

if (this.Text == "Employees")

{

this.dataGridView.Columns["Login"].ReadOnly = true;

this.dataGridView.Columns["Password"].Visible = false;

}

foreach (DataColumn col in dt.Columns)

searchColumnComboBox.Items.Add(col.Caption);

}

public void bindingNavigatorSaveItem\_Click(object sender, EventArgs e)

{

SQLHelper.Reseed(this.Text, (dataGridView.RowCount - 2));

SQLHelper.UpdateData(this.Text, ds);

toolStripProgressBar.Value = 100;

}

private void TableForm\_KeyDown(object sender, KeyEventArgs e)

{

if (e.Modifiers == Keys.Control && e.KeyCode == Keys.S) {

dataGridView.EndEdit();

bindingNavigatorSaveItem\_Click(null, null); }

if (e.Modifiers == Keys.Control && e.KeyCode == Keys.E)

dataGridView.BeginEdit(true);

if (e.KeyCode == Keys.Delete)

{

DialogResult result = MessageBox.Show("Confirm delete?", "Warning!",

MessageBoxButtons.YesNo, MessageBoxIcon.Warning,

MessageBoxDefaultButton.Button1);

if (result == DialogResult.Yes)

{

dataGridView.Rows.Remove(dataGridView.CurrentRow);

bindingNavigatorSaveItem\_Click(null, null);

}

else this.Focus();

}

}

public void comboClear()

{

variationComboBox.Items.Clear();

variationComboBox.Text = "";

}

private void dataGridView\_CurrentCellChanged(object sender, EventArgs e)

{

try

{

switch (this.Text)

{

case "Car\_catalog":

switch (dataGridView.CurrentCell.ColumnIndex)

{

case 1:

comboClear();

oldParentFolderName = dataGridView.CurrentCell.Value.ToString();

variationComboBox.Text = dataGridView.CurrentCell.Value.ToString();

variationComboBox.Items.AddRange(SQLHelper.Items("Name", "Brand\_name"));

break;

case 2:

oldFolderName = dataGridView.CurrentCell.Value.ToString();

break;

case 3:

comboClear();

variationComboBox.Text = dataGridView.CurrentCell.Value.ToString();

variationComboBox.Items.AddRange(SQLHelper.Items("Name", "Body\_type"));

break;

case 4:

comboClear();

variationComboBox.Text = dataGridView.CurrentCell.Value.ToString();

variationComboBox.Items.AddRange(SQLHelper.Items("Name", "Gearbox\_type"));

break;

case 5:

comboClear();

variationComboBox.Text = dataGridView.CurrentCell.Value.ToString();

variationComboBox.Items.AddRange(SQLHelper.Items("Name", "Engine\_type"));

break;

case 6:

comboClear();

variationComboBox.Text = dataGridView.CurrentCell.Value.ToString();

variationComboBox.Items.AddRange(SQLHelper.Items("Name", "Drive\_type"));

break;

case 7:

comboClear();

variationComboBox.Text = dataGridView.CurrentCell.Value.ToString();

variationComboBox.Items.Add("Left");

variationComboBox.Items.Add("Right");

break;

default: comboClear();

break;

}

break;

case "Orders":

switch (dataGridView.CurrentCell.ColumnIndex)

{

case 2:

comboClear();

variationComboBox.Text = dataGridView.CurrentCell.Value.ToString();

variationComboBox.Items.AddRange(SQLHelper.Items("Id", "Car\_catalog"));

break;

case 3:

comboClear();

variationComboBox.Text = dataGridView.CurrentCell.Value.ToString();

variationComboBox.Items.AddRange(SQLHelper.Items("Model","Car\_catalog"));

break;

case 4:

comboClear();

variationComboBox.Text = dataGridView.CurrentCell.Value.ToString();

variationComboBox.Items.AddRange(SQLHelper.Items("VIN", "Cars\_VINs"));

break;

case 5:

comboClear();

variationComboBox.Text = dataGridView.CurrentCell.Value.ToString();

variationComboBox.Items.AddRange(SQLHelper.Items("Name", "Order\_status\_type"));

break;

case 7:

comboClear();

variationComboBox.Text = dataGridView.CurrentCell.Value.ToString();

variationComboBox.Items.AddRange(SQLHelper.Items("Id", "Suppliers"));

break;

case 8:

comboClear();

variationComboBox.Text = dataGridView.CurrentCell.Value.ToString();

variationComboBox.Items.AddRange(SQLHelper.Items("Id", "Employees"));

break;

default: comboClear();

break;

}

break;

case "Leasing":

switch (dataGridView.CurrentCell.ColumnIndex)

{

case 1:

comboClear();

variationComboBox.Text = dataGridView.CurrentCell.Value.ToString();

variationComboBox.Items.AddRange(SQLHelper.Items("Id", "Orders"));

break;

case 2:

comboClear();

variationComboBox.Text = dataGridView.CurrentCell.Value.ToString();

variationComboBox.Items.Add("12");

variationComboBox.Items.Add("24");

variationComboBox.Items.Add("36");

break;

case 5:

comboClear();

variationComboBox.Text = dataGridView.CurrentCell.Value.ToString();

variationComboBox.Items.Add("Yes");

variationComboBox.Items.Add("No");

break;

default:

comboClear();

break;

}

break;

case "Employees":

if (dataGridView.CurrentCell.ColumnIndex == 1)

{

variationComboBox.Text = dataGridView.CurrentCell.Value.ToString();

variationComboBox.Items.Add("Saler");

variationComboBox.Items.Add("Director");

} else comboClear();

break;

case "Clients":

switch (dataGridView.CurrentCell.ColumnIndex) {

case 3:

comboClear();

variationComboBox.Text = dataGridView.CurrentCell.Value.ToString();

variationComboBox.Items.AddRange(SQLHelper.Items("Id", "Orders"));

break;

case 4:

comboClear();

variationComboBox.Text = dataGridView.CurrentCell.Value.ToString();

variationComboBox.Items.AddRange(SQLHelper.Items("Name", "Order\_status\_type"));

break;

case 5:

comboClear();

variationComboBox.Text = dataGridView.CurrentCell.Value.ToString();

variationComboBox.Items.AddRange(SQLHelper.Items("Id", "Cars\_VINs", "Car\_model\_id",

"Order\_id", new SqlParameter("@Car\_model\_id", dataGridView.CurrentRow.Cells[3]

.Value), new SqlParameter("@Order\_id", null)));

break;

case 6:

comboClear();

variationComboBox.Text = dataGridView.CurrentCell.Value.ToString();

variationComboBox.Items.AddRange(SQLHelper.Items("VIN", "Cars\_VINs", "Car\_model\_id",

"Order\_id", new SqlParameter("@Car\_model\_id", dataGridView.CurrentRow.Cells[3]

.Value), new SqlParameter("@Order\_id", null)));

break;

default : comboClear();

break;

}

break;

}

if (dataGridView.CurrentRow.Index > 0) toolStripProgressBar.Value = dt

.Rows[dataGridView.CurrentRow.Index - 1].RowState != DataRowState

.Unchanged ? 0 : 100;

}

catch (NullReferenceException) { return; }

}

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

{

try

{

if ((this.Text == "Clients") && (dataGridView.CurrentCell.ColumnIndex == 3))

using (SqlDataReader reader = SQLHelper.ExecuteReader("Status", "Orders", "Id",

new SqlParameter("@Id", variationComboBox.SelectedItem)))

{

reader.Read();

dataGridView.CurrentRow.Cells[4].Value = reader.GetValue(0);

reader.Close();

}

if (this.Text == "Orders")

if (dataGridView.CurrentCell.ColumnIndex == 2)

{

using (SqlDataReader reader = SQLHelper.ExecuteReader("Model, Cost", "Car\_catalog",

"Id", new SqlParameter("@Id", variationComboBox.SelectedItem)))

{

reader.Read();

if (string.IsNullOrEmpty(dataGridView.CurrentRow.Cells[3].Value.ToString()))

dataGridView.CurrentRow.Cells[3].Value = reader.GetString(0);

dataGridView.CurrentRow.Cells[6].Value = reader.GetValue(1).ToString();

reader.Close();

}

}

else if (dataGridView.CurrentCell.ColumnIndex == 3)

{

using (SqlDataReader reader = SQLHelper.ExecuteReader("Id, Cost", "Car\_catalog",

"Model", new SqlParameter("@Model", variationComboBox.SelectedItem)))

{

reader.Read();

if (dataGridView.CurrentRow.Cells[2] == null)

dataGridView.CurrentRow.Cells[2].Value = reader.GetValue(0).ToString();

dataGridView.CurrentRow.Cells[6].Value = reader.GetValue(1).ToString();

reader.Close();

}

}

if (this.Text == "Leasing")

if (dataGridView.CurrentCell.ColumnIndex == 1)

{

dataGridView.CurrentCell.Value = variationComboBox.SelectedItem;

if (!string.IsNullOrEmpty(dataGridView.CurrentRow.Cells[1].Value.ToString()))

using (SqlDataReader reader = SQLHelper.ExecuteReader("Cost",

"Orders", "Id", new SqlParameter("@Id", variationComboBox.SelectedItem)))

{

reader.Read();

dataGridView.CurrentRow.Cells[3].Value = reader.GetValue(0);

reader.Close();

}

}

else if (dataGridView.CurrentCell.ColumnIndex == 5)

dataGridView.CurrentCell.Value = variationComboBox.SelectedText == "Yes" ? 0 : 1;

dataGridView.CurrentCell.Value = variationComboBox.SelectedItem;

}

catch (NullReferenceException) { return; }

dataGridView.Focus();

}

private void searchTextBox\_TextChanged(object sender, EventArgs e)

{

if (searchColumnComboBox.SelectedItem == null) MessageBox.Show("Select filtering column");

else bs.Filter = String.Format(searchColumnComboBox.SelectedItem + " like '%"

+ searchTextBox.Text + "%'");

}

private void dataGridView\_CellEndEdit(object sender, DataGridViewCellEventArgs e)

{

if ((this.Text == "Car\_catalog"))

{

if (e.ColumnIndex == 2)

{

string resPath = Path.GetFullPath("Resources");

if (!Directory.Exists(oldFolderName))

{

OpenFileDialog ofd = new OpenFileDialog();

ofd.Title = "Add pictures";

ofd.ShowDialog();

string brandFolder = resPath + "\\" + dataGridView.CurrentRow.Cells[1].Value.ToString();

if (!Directory.Exists(brandFolder)) Directory.CreateDirectory(brandFolder);

string picFolderPath = brandFolder + "\\" + dataGridView.CurrentCell.Value.ToString();

if (!Directory.Exists(picFolderPath)) Directory.CreateDirectory(picFolderPath);

try

{

File.Copy(picFolderPath, ofd.FileName, true);

} catch (Exception ex) { MessageBox.Show(ex.Message); return; }

}

else

{

string oldPath = resPath + "\\" + oldParentFolderName + "\\" + oldFolderName;

if (Directory.Exists(oldPath))

{

string newPath = resPath + "\\" + dataGridView.CurrentRow.Cells[1].Value.ToString()

+ "\\" + dataGridView.CurrentCell.Value.ToString();

Directory.CreateDirectory(newPath);

foreach (string dir in Directory.GetFiles(oldPath))

Directory.Move(dir, newPath);

Directory.Delete(oldPath, true);

}

else { MessageBox.Show("Directiry not exist!", "Warning", MessageBoxButtons.OK,

MessageBoxIcon.Error); }

}

}

if (RowExist(dataGridView.CurrentRow))

{

MessageBox.Show(RowExist(dataGridView.CurrentRow).ToString());

dataGridView.Rows.Remove(dataGridView.CurrentRow);

}

}

if ((this.Text == "Employees") && (e.ColumnIndex == 5)

&& (dataGridView.CurrentRow.Cells[6].Value.ToString() == ""))

{

bindingNavigatorSaveItem\_Click(null, null);

LoginForm lf = new LoginForm();

lf.Text = "Register";

lf.Show();

}

}

public bool RowExist(DataGridViewRow e)

{

try

{

int counter = 0;

for (int i = 0; i < dataGridView.RowCount - 2; i++)

{

for (int j = 0; j < dataGridView.ColumnCount; j++)

{

if (e.Cells[j].Value.ToString() == dataGridView.Rows[i].Cells[j].Value.ToString()) counter++;

}

if (counter >= 13) return true;

}

return false;

} catch (IndexOutOfRangeException) { return false; }

}

Word.\_Application wordApp;

Word.\_Document doc = null;

string appPath = System.Windows.Forms.Application.StartupPath;

object miss = Type.Missing;

public void Replace(string mark, string value)

{

doc.Bookmarks[mark].Range.Text = value;

}

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

{

object fileName = null;

if (this.Text == "Orders") fileName = appPath + "\\Docs\\Agreement.docx";

else if (this.Text == "Leasing") fileName = appPath + "\\Docs\\LeasingAgreement.dotx";

wordApp = new Word.Application();

try

{

doc = wordApp.Documents.Open(ref fileName, ref miss, true, ref miss, ref miss, ref miss,

ref miss, ref miss, ref miss, ref miss, ref miss, true, ref miss, ref miss,

ref miss, ref miss);

}

catch (Exception ex)

{

doc.Close(false, ref miss, ref miss);

wordApp.Quit(ref miss, ref miss, ref miss);

doc = null;

wordApp = null;

throw ex;

}

if (this.Text == "Leasing")

{

string DirFullName = SQLHelper.ExecuteReaderFrom("Full\_Name", "Employees", "Position",

new SqlParameter("@Position", "Director")).ToString();

string[] DirFullNameSplit = DirFullName.Split(new char[] { ' ' }, StringSplitOptions.

RemoveEmptyEntries);

Replace("DirSurname", DirFullNameSplit[0]);

Replace("DirName", DirFullNameSplit[1]);

Replace("DirPatronymic", DirFullNameSplit[2]);

string ClientName = SQLHelper.ExecuteReaderFrom("Full\_Name", "Clients", "Order\_id",

new SqlParameter("@Order\_id", dataGridView.CurrentRow.Cells[1].Value)).ToString();

Replace("Name", ClientName);

string country = SQLHelper.ExecuteReaderFrom("Country", "Brand\_name", "Id",

new SqlParameter("@Id", SQLHelper.ExecuteReaderFrom("Id", "Orders",

"Car\_model\_id", new SqlParameter("@Car\_model\_id", SQLHelper

.ExecuteReaderFrom("Car\_model\_id", "Orders", "Id", new SqlParameter("@Id",

dataGridView.CurrentRow.Cells[1].Value)))))).ToString();

Replace("Country", country);

Replace("Cost", dataGridView.CurrentRow.Cells[3].Value.ToString());

Replace("TotalCost", dataGridView.CurrentRow.Cells[3].Value.ToString());

string EmpInfo = null;

try

{

using (SqlDataReader reader = SQLHelper.ExecuteReader("Position, Full\_Name, Phone, Address",

"Employees", "Id", new SqlParameter("@Id", Convert.ToInt32(SQLHelper.ExecuteReaderFrom("Issued\_by",

"Orders", "Id", new SqlParameter("@Id", dataGridView.CurrentRow.Cells[1].Value))))))

{

reader.Read();

for (int i = 0; i < reader.FieldCount; i++)

{

EmpInfo += i < reader.FieldCount - 1 ? (reader.GetString(i) + ", ") : reader.GetString(i);

}

reader.Close();

}

}

catch (Exception ex) { MessageBox.Show(ex.Message); }

Replace("EmpInfo", EmpInfo);

DateTime date = Convert.ToDateTime(SQLHelper.ExecuteReaderFrom("Date", "Orders", "Id",

new SqlParameter("@Id", dataGridView.CurrentRow.Cells[1].Value)));

Replace("SupDate", date.Date.ToString());

Replace("Term", dataGridView.CurrentRow.Cells[4].Value.ToString());

Replace("Prepay", dataGridView.CurrentRow.Cells[6].Value.ToString());

Replace("Contact", EmpInfo);

}

wordApp.Visible = true;

if (wordApp.Visible == false) wordApp.Quit(ref miss, ref miss, ref miss);

}

Excel.Application excelApp = null;

Excel.Workbook book = null;

Excel.Worksheet sheet = null;

public void SetValue(object value, int rowIndex, int colIndex)

{

sheet.Cells[rowIndex, colIndex] = value;

}

public void AddChart()

{

Excel.Chart chart = (Excel.Chart)excelApp.Charts.Add(miss, miss, miss, miss);

chart.Activate();

chart.Select(miss);

}

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

{

excelApp = new Excel.Application();

book = excelApp.Workbooks.Add(miss);

try {

sheet = (Excel.Worksheet)book.Worksheets[1];

}

catch (System.Runtime.InteropServices.COMException ex)

{

MessageBox.Show(ex.Message);

if (ex.Message == "Исключение из HRESULT: 0x80010001 (RPC\_E\_CALL\_REJECTED)")

Process.Start("https://support.microsoft.com/ru-ru/topic/появляется-сообщение-об-ошибке" +

"-или-программа-перестает-отвечать-на-запросы-в-профессиональном-учете-или-в-экспре" +

"сс-fbfad9f7-d574-598d-54c6-5beef60f0984");

}

//int rowIndex = dataGridView.Rows.Count;

if (this.Text == "Orders")

{

//Get date

DateTime date = Convert.ToDateTime(dataGridView.CurrentRow.Cells["Date"].Value);

//Add Columns names

SetValue("Brand", 1, 1);

SetValue("Model", 1, 2);

SetValue("Cost", 1, 3);

SetValue("Count", 1, 4);

SetValue("Leasing", 1, 5);

SetValue("NonLeasing", 1, 6);

SetValue("Revenue", 1, 7);

string brand = null;

string[] models = null;

Excel.Range cells = null;

int r = 2;

int r1 = 2;

//Get models catalog

string[,] mas = SQLHelper.AllItems("Brand, Id, Model, Cost", "Car\_catalog");

SetValue(mas[0, 0], r1, 1);

models = new string[mas.GetLength(0)];

models[0] = mas[0, 2];

for (int i = 0; i < mas.GetLength(0); i++)

{

if ((i > 0) && (mas[i, 0] != brand))

{

SetValue(mas[i, 0], r1, 1);

brand = mas[i, 0];

if (r1 - r > 1)

{

cells = (Excel.Range)sheet.get\_Range("A" + r, "A" + r1);

cells.Merge(miss);

}

r = r1;

}

SetValue(mas[i, 2], r1, 2);

models[i] = mas[i, 2];

SetValue(mas[i, 3].Remove(mas[i, 3].Length - 5, 5), r1, 3);

int count, leas;

count = SQLHelper.Count("Id", "Orders", "Car\_model\_id", "Date",

new SqlParameter("@Car\_model\_id", mas[i, 1]),

new SqlParameter("@Date", date));

SetValue(count, r1, 4);

leas = SQLHelper.Count("Id", "Leasing", "Model\_id", "Order\_date",

new SqlParameter("@Model\_id", mas[i, 1]),

new SqlParameter("@Order\_date", date));

SetValue(leas, r1, 5);

SetValue((count - leas), r1, 6);

double formula = (1.1 \*leas + count) \* Convert.ToDouble(mas[i, 3].Remove(mas[i, 3]

.Length - 5, 5));

sheet.Cells[r1, 7].Formula = (formula);

//SetValue(formula, r1, 7);

r1++;

}

//Diagramms

//Sold count histogramm

AddHistogramm(cells, date, models, r1);

AddPie(cells, date, models, r1);

}

excelApp.Visible = true;

if (excelApp.Visible == false) excelApp.Quit();

}

public void AddHistogramm(Excel.Range cells, DateTime date, string[] models, int r1)

{

//Select range of cells

cells = (Excel.Range)sheet.get\_Range("D2", "D" + (r1 - 1));

cells.Select();

//Add chart

AddChart();

//Change type of chart

excelApp.ActiveChart.ChartType = Excel.XlChartType.xlColumnStacked;

excelApp.ActiveChart.HasTitle = true;

//Add title text

excelApp.ActiveChart.ChartTitle.Text = "Count of purchosed car on " + date.Month +

"th month of " + date.Year;

//Add bottom axis

((Excel.Axis)excelApp.ActiveChart.Axes(Excel.XlAxisType.xlCategory, Excel.XlAxisGroup

.xlPrimary)).HasTitle = true;

((Excel.Axis)excelApp.ActiveChart.Axes(Excel.XlAxisType.xlCategory, Excel.XlAxisGroup

.xlPrimary)).AxisTitle.Text = "Model";

//Add left axis

((Excel.Axis)excelApp.ActiveChart.Axes(Excel.XlAxisType.xlValue, Excel.XlAxisGroup

.xlPrimary)).HasTitle = true;

((Excel.Axis)excelApp.ActiveChart.Axes(Excel.XlAxisType.xlValue, Excel.XlAxisGroup

.xlPrimary)).AxisTitle.Text = "Quanity";

//Add grid lines of axis

((Excel.Axis)excelApp.ActiveChart.Axes(Excel.XlAxisType.xlCategory, Excel.XlAxisGroup

.xlPrimary)).HasMajorGridlines = true;

((Excel.Axis)excelApp.ActiveChart.Axes(Excel.XlAxisType.xlCategory, Excel.XlAxisGroup

.xlPrimary)).HasMinorGridlines = true;

((Excel.Axis)excelApp.ActiveChart.Axes(Excel.XlAxisType.xlValue, Excel.XlAxisGroup

.xlPrimary)).HasMajorGridlines = true;

((Excel.Axis)excelApp.ActiveChart.Axes(Excel.XlAxisType.xlValue, Excel.XlAxisGroup

.xlPrimary)).HasMinorGridlines = true;

//Add chart legend

excelApp.ActiveChart.HasLegend = true;

//Change legend position

excelApp.ActiveChart.Legend.Position = Excel.XlLegendPosition.xlLegendPositionBottom;

Excel.SeriesCollection seriesCollection = (Excel.SeriesCollection)excelApp.ActiveChart

.SeriesCollection(miss);

//Delete default legend data list

((Excel.LegendEntry)excelApp.ActiveChart.Legend.LegendEntries(1)).Delete();

//Add series names

Excel.Series series = seriesCollection.Item(1);

series.Name = "Model";

series.XValues = models as Array;

//Add chart to list

excelApp.ActiveChart.Location(Excel.XlChartLocation.xlLocationAsObject, "Лист1");

sheet = (Excel.Worksheet)excelApp.Worksheets.get\_Item(1);

//Change chart position

sheet.Shapes.Item(1).IncrementLeft(200);

sheet.Shapes.Item(1).IncrementTop(-120);

//Change chart size

sheet.Shapes.Item(1).Height = 300;

sheet.Shapes.Item(1).Width = 350;

}

public void AddPie(Excel.Range cells, DateTime date, string[] models, int r1)

{

//Select range of cells

cells = (Excel.Range)sheet.get\_Range("G2", "G" + (r1 - 1));

cells.Select();

//Add chart

AddChart();

//Change type of chart

excelApp.ActiveChart.ChartType = Excel.XlChartType.xlPie;

excelApp.ActiveChart.HasTitle = true;

//Add title text

excelApp.ActiveChart.ChartTitle.Text = "Revenue on " + date.Month + "th month";

//Add chart legend

excelApp.ActiveChart.HasLegend = true;

//Change legend position

excelApp.ActiveChart.Legend.Position = Excel.XlLegendPosition.xlLegendPositionRight;

Excel.SeriesCollection seriesCollection = (Excel.SeriesCollection)excelApp.ActiveChart

.SeriesCollection(miss);

//Add series names

Excel.Series series = seriesCollection.Item(1);

series.Name = "Revenue";

series.XValues = models as Array;

//Add chart to list

excelApp.ActiveChart.Location(Excel.XlChartLocation.xlLocationAsObject, "Лист1");

sheet = (Excel.Worksheet)excelApp.Worksheets.get\_Item(1);

//Change chart position

sheet.Shapes.Item(2).IncrementLeft(200);

sheet.Shapes.Item(2).IncrementTop(200);

//Change chart size

sheet.Shapes.Item(2).Height = 300;

sheet.Shapes.Item(2).Width = 350;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace CarInfo

{

internal static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new LoginForm());

Application.Run(new CarInfoForm());

}

}

}

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace CarInfo

{

public class SQLHelper

{

static string location = System.Reflection.Assembly.GetExecutingAssembly().Location;

static string path = System.IO.Path.GetDirectoryName(location);

public static SqlConnection GetConn()

{

SqlConnection conn = new SqlConnection("Data Source=(LocalDB)\\MSSQLLocalDB;" +

"AttachDbFilename=" + path + "\\Car\_directory.mdf;Integrated Security=True;" +

"Connect Timeout=30");

return conn;

}

public static void Reseed(string table, int index)

{

SqlConnection con = GetConn();

index = index < 0 ? 0 : index;

string query = "DBCC CHECKIDENT ('" + table + "', RESEED, " + index + ")";

using (SqlCommand cmd = new SqlCommand(query, con))

{

cmd.CommandText = query;

con.Open();

cmd.ExecuteNonQuery();

}

con.Close();

}

public static SqlDataReader ExecuteReader(string col, string table, string selector,

SqlParameter parameter)

{

SqlConnection con = GetConn();

string query = "SELECT " + col + " FROM " + table + " WHERE " + selector +

" = @" + selector;

using (SqlCommand cmd = new SqlCommand(query, con))

{

cmd.CommandText = query;

cmd.Parameters.Add(parameter);

con.Open();

SqlDataReader reader = cmd.ExecuteReader();

return reader;

}

}

public static object ExecuteReaderFrom(string col, string table, string selector,

SqlParameter parameter)

{

object res = null;

SqlConnection con = GetConn();

string query = "SELECT " + col + " FROM " + table + " WHERE " + selector +

" = @" + selector;

using (SqlCommand cmd = new SqlCommand(query, con))

{

cmd.CommandText = query;

cmd.Parameters.Add(parameter);

con.Open();

SqlDataReader reader = cmd.ExecuteReader();

try

{

reader.Read();

res = reader.GetValue(0);

reader.Close();

} catch (InvalidOperationException ex) { MessageBox.Show(ex.Message); return null; }

con.Close();

}

return res;

}

public static string[] Items(string col, string table)

{

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

string[] itms;

SqlConnection con = GetConn();

string query = "SELECT " + col + " FROM " + table;

using (SqlCommand cmd = new SqlCommand(query, con))

{

cmd.CommandText = query;

con.Open();

SqlDataReader reader = cmd.ExecuteReader();

while (reader.Read()) list.Add(reader.GetValue(0).ToString());

itms = list.ToArray();

reader.Close();

con.Close();

}

return itms;

}

public static string[,] AllItems(string col, string table)

{

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

string[] qlist;

string[,] itms;

SqlConnection con = GetConn();

string query = "SELECT " + col + " FROM " + table;

using (SqlCommand cmd = new SqlCommand(query, con))

{

cmd.CommandText = query;

con.Open();

SqlDataReader reader = cmd.ExecuteReader();

while (reader.Read())

for (int i = 0; i < reader.FieldCount; i++)

{

list.Add(reader.GetValue(i).ToString());

}

qlist = list.ToArray();

int q = reader.FieldCount;

itms = new string[(qlist.Length + 1) / q, q];

int r = 0, c = 0;

for (int i = 0; i < qlist.Length; i++)

{

if ((i > 0) && (i % q == 0)) { r++; c = 0; }

itms[r,c] = qlist[i];

c++;

}

reader.Close();

con.Close();

}

return itms;

}

public static string[] Items(string col, string table, string selector1,

string selector2, SqlParameter parameter1, SqlParameter parameter2)

{

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

string[] itms;

SqlConnection con = GetConn();

string query = "SELECT " + col + " FROM " + table + " WHERE " + selector1 +

" = @" + selector1 + " AND " + selector2 + " = @" + selector2;

using (SqlCommand cmd = new SqlCommand(query, con))

{

cmd.CommandText = query;

cmd.Parameters.Add(parameter1);

cmd.Parameters.Add(parameter2);

con.Open();

try

{

SqlDataReader reader = cmd.ExecuteReader();

while (reader.Read()) list.Add(reader.GetValue(0).ToString());

itms = list.ToArray();

reader.Close();

}

catch (Exception ex) { MessageBox.Show(ex.Message); return null; }

con.Close();

}

return itms;

}

public static int Count(string col, string table, string selector1,

string selector2, SqlParameter parameter1, SqlParameter parameter2)

{

int res;

SqlConnection con = GetConn();

string query = "SELECT COUNT(" + col + ") FROM " + table + " WHERE " + selector1 +

" = @" + selector1 + " AND MONTH(" + selector2 + ") = MONTH(@" + selector2 + ")" +

"AND YEAR(" + selector2 + ") = YEAR(@" + selector2 + ")";

using (SqlCommand cmd = new SqlCommand(query, con))

{

cmd.CommandText = query;

cmd.Parameters.Add(parameter1);

cmd.Parameters.Add(parameter2);

con.Open();

SqlDataReader reader = cmd.ExecuteReader();

reader.Read();

res = reader.GetInt32(0);

reader.Close();

con.Close();

}

return res;

}

public static DataSet LoadData(string table)

{

SqlConnection con = GetConn();

con.Open();

string query = "SELECT \* FROM " + table;

SqlDataAdapter adapter = new SqlDataAdapter(query, con);

DataSet ds = new DataSet();

adapter.Fill(ds, table);

adapter.Dispose();

con.Close();

return ds;

}

public static DataSet UpdateData(string table, DataSet ds)

{

SqlConnection con = GetConn();

con.Open();

string query = "SELECT \* FROM " + table;

SqlDataAdapter adapter = new SqlDataAdapter(query, con);

SqlCommandBuilder cmdBuilder = new SqlCommandBuilder(adapter);

try

{

adapter.Update(ds, table);

}

catch (Exception e) { MessageBox.Show(e.Message); }

ds.Clear();

adapter.Fill(ds, table);

con.Close();

return ds;

}

public static void GenVINS(string id)

{

SqlConnection con = GetConn();

string[] yearMas = {"A", "B", "C", "D", "E", "F", "G", "H", "J", "K", "L", "M", "N", "P",

"R", "S", "T", "V", "W", "X", "Y", "1", "2", "3", "4", "5", "6", "7", "8", "9"};

using (SqlDataReader reader = ExecuteReader("\*", "Car\_catalog",

"Id = @Id", new SqlParameter("@Id", id)))

{

string WMI = ExecuteReaderFrom("WMI", "Brand\_name", "Name",

new SqlParameter("@Name", reader.GetString(3))).ToString() + "9";

string VDS = reader.GetString(3).Split(new char[] { ' ' },

StringSplitOptions.RemoveEmptyEntries)[0] + ExecuteReaderFrom("Id", "Body\_type",

"Name", new SqlParameter("@Name", reader.GetString(3))).ToString() + 1 +

ExecuteReaderFrom("Id", "Engine\_type", "Name", new SqlParameter("@Name",

reader.GetString(5))).ToString() + "00";

int year = Convert.ToInt32(reader.GetDateTime(13).Year);

string VIS = yearMas[year - (year % 30) \* 30] +

"0" + WMI + "1";

string VIN = WMI + VDS + VIS;

string query = "INSERT INTO Cars\_VINs (VIN, Car\_model\_id, Car\_model) VALUES ";

for (int i = 0; i < 40; i++)

{

if (i % 1000 >= 0) VIN += i;

else if (i % 100 >= 0) VIN += "0" + i;

else if (i % 10 >= 0) VIN +="00" + i;

else VIN += "000" + i;

query += '(' + VIN + ',' + reader.GetInt32(0).ToString() + ',' +

reader.GetString(2) + ')';

using (SqlCommand cmd = new SqlCommand(query, con))

{

cmd.CommandText = query;

con.Open();

cmd.ExecuteNonQuery();

con.Close();

}

}

}

}

}

}

# ПРИЛОЖЕНИЕ 2

# БЛОК-СХЕМА АЛГОРИТМА РАБОТЫ ПРОГРАММНОГО СРЕДСТВА

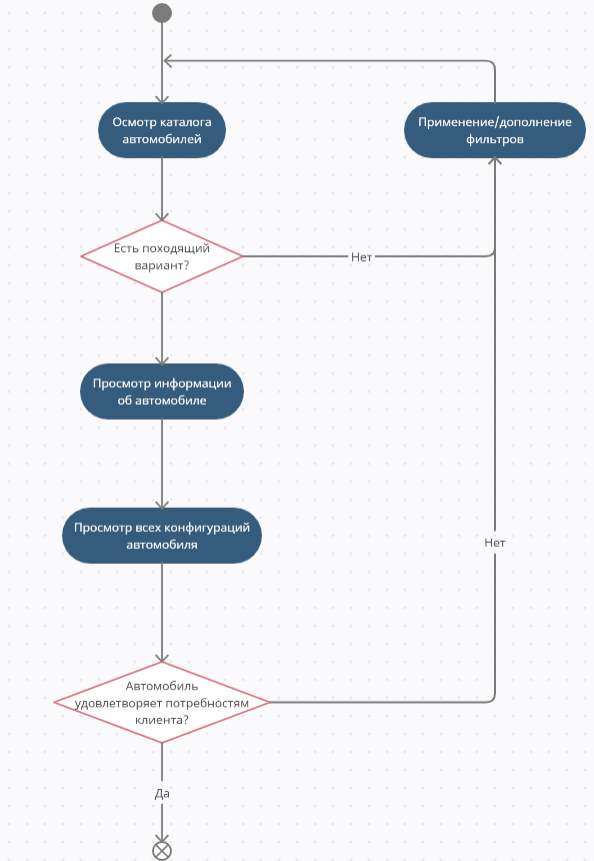


Рисунок П2.1 – Блок-схема работы программного средства

# ПРИЛОЖЕНИЕ 3

# РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Для запуска приложения необходимо запустить .exe файл (рис. П3.1) в корневой папке приложения.



Рисунок П3.1 - .exe файл приложения

При запуске приложения появиться окно входа (рис. П3.2), вводим логин и пароль в соответствующие поля. Вход осуществляется кнопкой «Login», а также нажатием клавиши Enter в поле пароля. В случае закрытия окна входа приложение так же закроется.

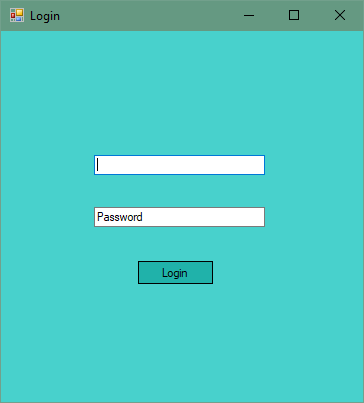


Рисунок П3.2 – Окно входа

После успешного входа окно логина закрывается и открывается главное окно приложения (рис. П3.3). В шапке данного окна расположены кнопки вызова вкладок. При нажатии любой из кнопок появляется дочернее окно, в случае, если нужное вам окно скрыто за другим нажмите на кнопку соответствующую вкладке, чтобы окно отобразилось поверх остальных. Для более удобной работы, можно воспользоваться группировкой окно, выбрав тип группировки из выпадающего меню кнопки «Window».

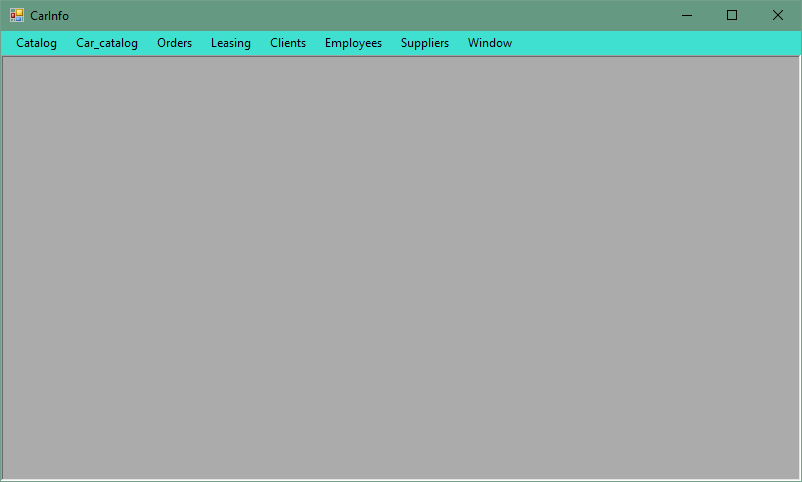


Рисунок П3.3 – Главное окно приложения

Кнопка «Catalog» отобразит окно каталога автомобилей (рис. П3.4). При наведении мышкой на фотографию автомобиля отображается марка автомобиля.

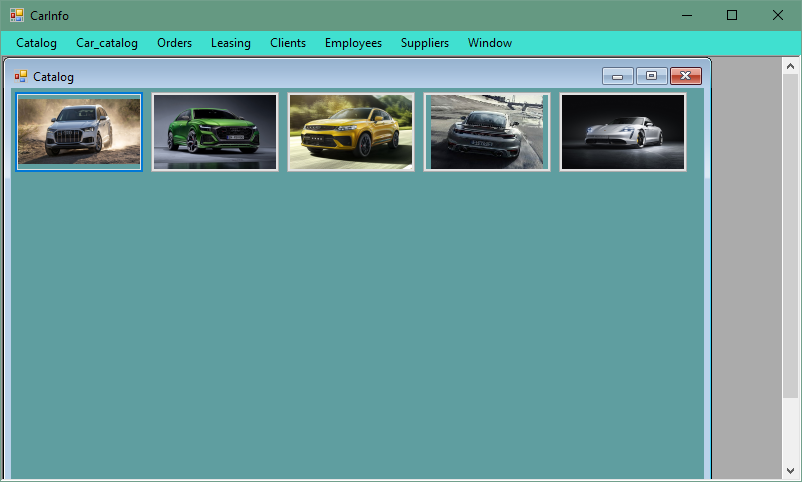


Рисунок П3.4 – Вкладка «Catalog»

При нажатии на картинку откроется окно характеристик соответствующего автомобиля (рис. П3.5).

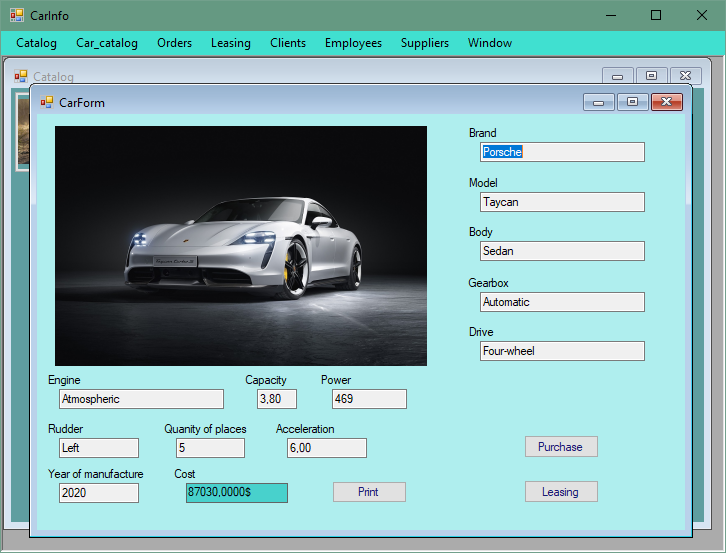


Рисунок П3.5 – Окно характеристик автомобиля

На окне характеристик имеется область с фотографиями, при нажатии на нее картинка меняется на следующую. Справа и снизу от этой области размещены характеристики автомобиля, поле со стоимостью автомобиля окрашено в бирюзовый цвет. Правее поля со стоимостью автомобиля находиться кнопка печати характеристик автомобиля, при нажатии на нее, откроется Word документ (рис. П3.6) c изображением и таблицей с характеристиками автомобиля.

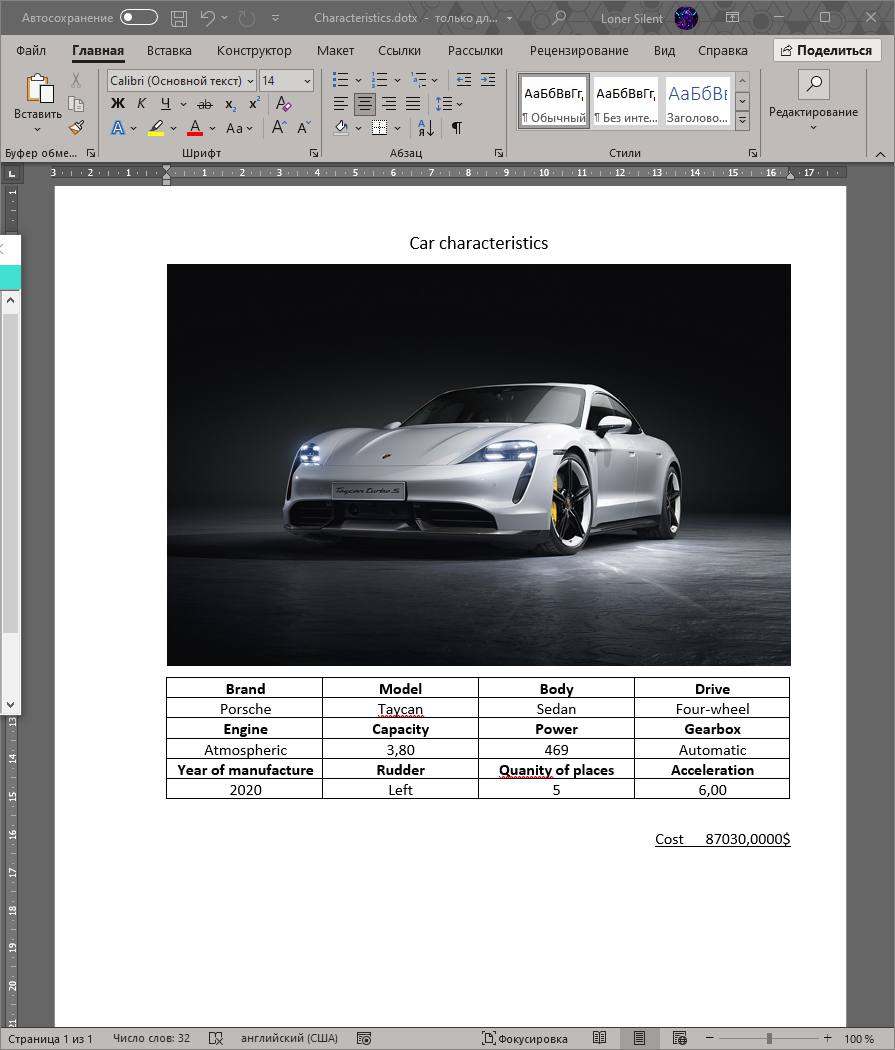


Рисунок П3.6 – Печать характеристик автомобиля

Остальные вкладки содержат таблицы с соответствующим их названию содержанием, поэтому рассмотрим работу с данными на примере вкладки «Orders» (рис. П3.7).

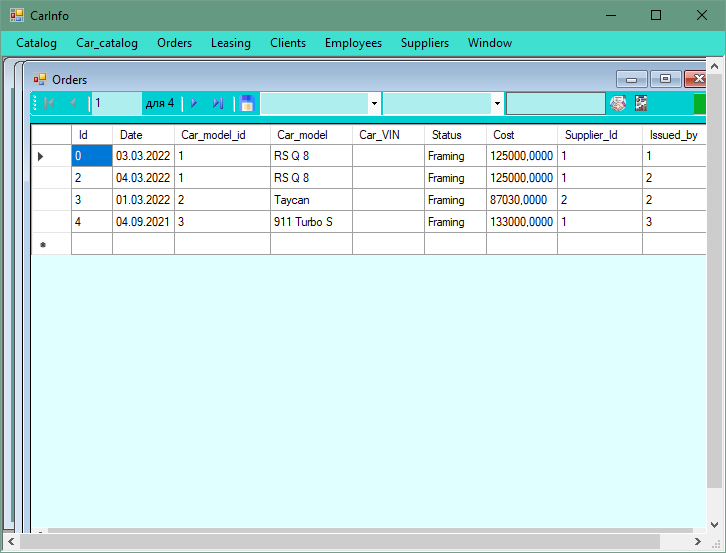


Рисунок П3.6 – Вкладка «Orders»

На каждой табличной вкладке помимо самой таблицы имеется панель элементов управления, содержащая весь необходимый функционал. Перемещаться по таблице можно с помощью синих треугольных кнопок, с помощью мышки или же при помощи клавиш стрелок («Вверх», «Вниз», «Вправо», «Влево»). Правее от кнопок перемещения «Вправо» находится кнопка сохранения «Save» (рис. П3.7), при нажатии на нее сохраняются изменения данных таблицы. !ВНИМАНИЕ : строка с измененными данными сохраняется только тогда, когда в любой ее ячейке редактирование завершено на клавишу «Enter», в противном случае все изменения пропадут! Также инициализация сохранения происходит при нажатии комбинации клавиш «Ctrl+S», для редактирования текущей ячейки существует комбинация клавиш «Ctrl+E». Далее после кнопки сохранения идет поле со стандартными вариантами значений в текущей ячейке (рис П3.8). Следующим идет поле для выбора колонки фильтра (рис П3.9), а после поле поиска (рис. П3.10). В самом правом углу находиться индикатор сохранения (рис П3.11), если он закрашен зеленым – данные сохранены, в ином случае изменения не сохраняться.



Рисунок П3.7 – Кнопка «Save»



Рисунок П3.8 – Поле стандартных вариантов



Рисунок П3.9 – Поле выбора колонки фильтра



Рисунок П3.10 – Поле поиска



Рисунок П3.11 – Индикатор сохранения

Это все стандартные элементы вкладок с таблицами, однако на вкладке «Orders» есть еще 2 кнопки: «Agreement» (рис. П3.12) и «Report» (рис. П3.13). При нажатии кнопки «Agreement» на вкладке «Orders» откроется шаблон Word документа купли-продажи автомобиля (рис. П3.14), который заполняется вручную, при нажатии этой же кнопки на вкладке «Leasing», откроется автоматически заполняемый шаблон Word документа лизингополучателя (рис. П3.15), однако некоторые поля все же придется заполнить вручную. Данная кнопка присутствует только на вкладках «Orders» и «Leasing», касательно кнопки «Report» - она доступна только на вкладке «Orders», и выполняет функцию создания Excel отчета (рис. П3.16) за месяц, соответствующий дате выбранной записи.



Рисунок П3.12 – Кнопка «Agreement»



Рисунок П3.13 – Кнопка «Report»

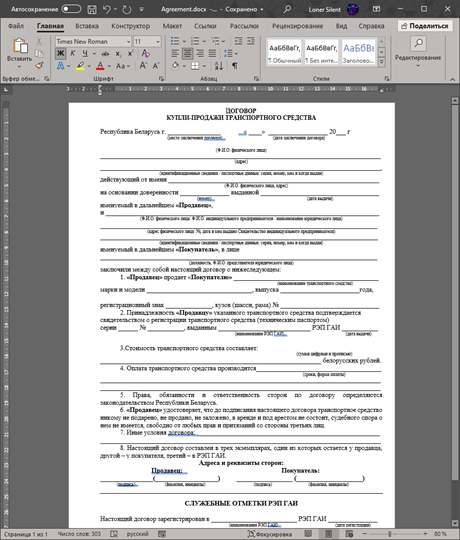


Рисунок П3.14 – Документ купли-продажи

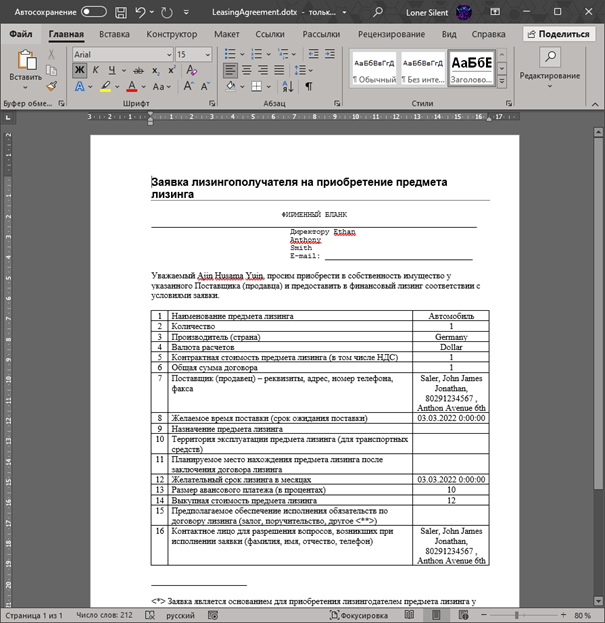


Рисунок П3.15 – Документ лизингополучателя

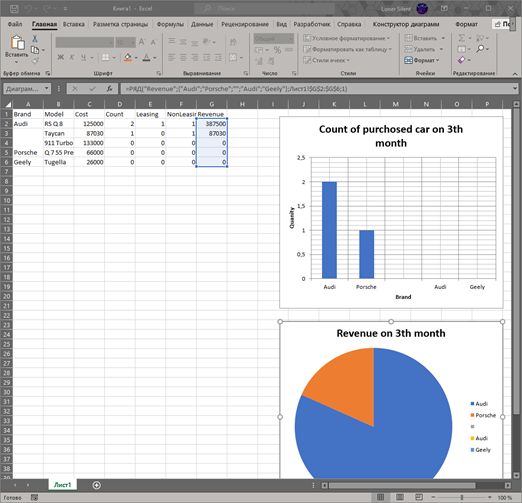


Рисунок П3.16 – Отчет за выбранный месяц

Стоит отметить, что при переходе из столбца «Model» в столбец «Body» будет выскакивать окно добавления картинок автомобиля (рис. П3.17), выбираем нужные картинки и нажимаем «Ок» чтобы их добавить. Также можно добавить картинки в папку «Resources» в корневой папке приложения. Разработчик советует добавлять картинки размером 1200 на 800.

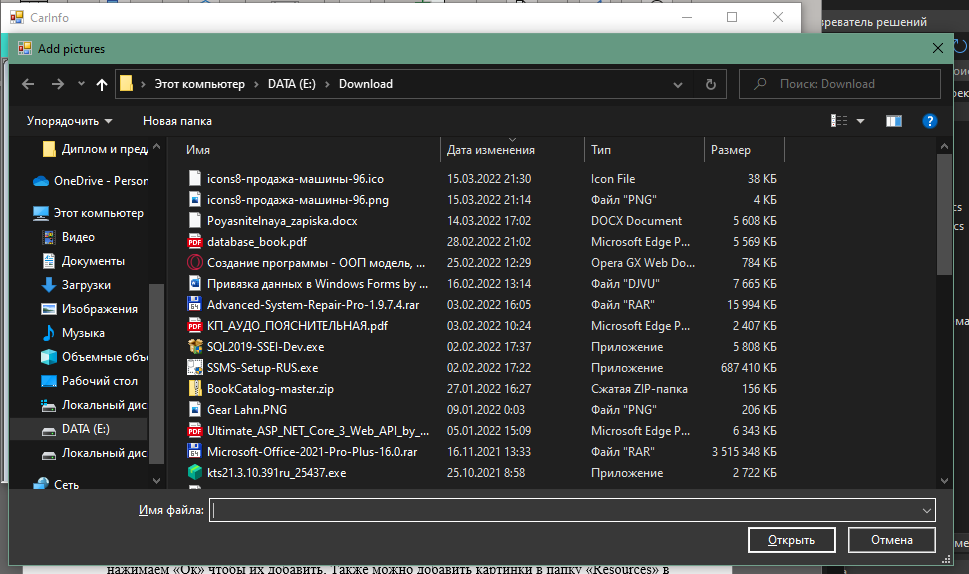


Рисунок П3.17

# ПРИЛОЖЕНИЕ 4

# ПРОТОКОЛ ИСПЫТАНИЙ ПРОГРАММНОГО СРЕДСТВА

Комиссия в составе:

* руководитель дипломного проекта – Растопчина Н.И.
* преподаватель дисциплин специального цикла – Разумов С.М.
* председатель цикловой комиссии – Тыбербай С.Г.

произвела проверку работоспособности программного средства, разработанного в порядке выполнения дипломного проекта.

|  |  |
| --- | --- |
| 1. Информация о программном средстве | |
| 1.1 Название программного средства | «Информационное приложение для автолюбителя» |
| 1.2 Номер версии программного средства | 1.04 |
| 1.3 Дата разработки версии программного средства | 20.02.2022 |
| 1.4 Назначение программного средства | Информационное приложение |
| 1.5 Функции программного средства | Информационное приложение для автолюбителя – программное средство с базой данных салона автомобилей.  Основная предназначение приложения: упростить работу сотрудников автомобильного салона по учету автомобилей и повысить их производительность. |
| 1.6 Язык программирования | C# |
| 1.7 Наличие программы инсталляции | Да |
| 1.8 Тип компьютера(ов) | ПК или ноутбук |
| 1.9 Требования к компьютеру:   * Процессор * Оперативная память (ОЗУ) * Место на диске | Intel Pentium III или выше  512 Mb  100 MB |
| 1.10 Дополнительное периферийное оборудование: принтеры, мышь, модем и т.п. | Клавиатура, компьютерная мышь |
| 1.11 Тип операционной системы (систем) | Windows 7 и выше |

|  |  |
| --- | --- |
| 2. Информация об организации-заказчике | |
| 2.1 Полное официальное наименование | Учреждение образования  «Витебский государственный колледж электротехники» |
| 2.2 Место нахождения (почтовый адрес) | г. Витебск, пр-т Черняховского, 14 |
| 2.3 Номер контактного телефона(ов) | 64-89-16 |
| 2.4 Номер факса | 64-89-16 |
| 2.5 Адрес электронной почты |  |
| 3. Информация о разработчике программного средства | |
| 3.1 Полное официальное наименование | Учреждение образования  «Витебский государственный колледж электротехники» |
| 3.2 Место нахождения (почтовый адрес) | г. Витебск, 6-я Луческая улица, д. 16 |
| 3.3 Фамилия, инициалы учащегося | Василевич М.И. |
| 3.4 Номер контактного телефона(ов) | +375 (29) 717-77-68 |
| 3.5 Номер факса | +375 (29) 717-77-68 |
| 3.6 Адрес электронной почты | Eyesless1jack@gmail.com |

Работоспособность программного средства проверена на контрольных примерах, предусматривающих все возможные варианты исходных и промежуточных данных.

Программное средство признано работоспособным, соответствующим техническому заданию и позволяет решать поставленные перед ним производственные задачи.

Руководитель дипломного проекта Растопчина Н.И.

Преподаватель дисциплин специального цикла Разумов С.М.

Председатель цикловой комиссии Тыбербай С.Г.