**Содержание**

[Введение 1](#_Toc149987071)

[1. Назначение и область применения программного продукта 2](#_Toc149987072)

[2. Постановка задачи 4](#_Toc149987073)

[2. 1 Определение требований к программной системе 4](#_Toc149987074)

[2. 2 Выбор и обоснование средств реализации 5](#_Toc149987075)

[3. Описание структуры ПС 7](#_Toc149987076)

[3. 1 Структура входных и выходных данных 7](#_Toc149987075)

[3. 2 Описание процедур и функций 7](#_Toc149987075)

[4. Проектирование 9](#_Toc149987077)

[4. 1 Структура БД 9](#_Toc149987075)

[4. 2 Описание решений по архитектуре 10](#_Toc149987075)

[4. 3 Проектирование интерфейса 18](#_Toc149987075)

[5. Описание интерфейса программного продукта 20](#_Toc149987078)

[6. Шаблоны проектирования 28](#_Toc149987079)

[7. Тестирование программного продукта 30](#_Toc149987080)

[Заключение 34](#_Toc149987081)

[Список использованной литературы 36](#_Toc149987082)

[Приложение А 37](#_Toc149987083)

# Введение

В современном мире прокат автомобилей является востребованной услугой, предоставляемой множеством предприятий. Однако эффективное управление данным бизнесом требует аккуратного контроля над клиентской базой данных, состоянием автомобилей, договорами аренды и ценовой политикой. Для обеспечения удобного и надёжного управления всеми этими аспектами разработано приложение для предприятия по прокату автомобилей.

Данное приложение предоставляет множество функций для эффективной работы с базами данных клиентов, автомобилей, состояний автомобилей, договоров аренды и цен на прокат. Кроме того, оно позволяет легко узнать, какие автомобили доступны для аренды в данный момент, а также изменить статус автомобиля со «свободен» на «арендован» и наоборот.

Наше приложение оснащено интуитивно понятным интерфейсом и удобными функциями, что облегчает управление предприятием по прокату автомобилей и ускоряет выполнение административных задач. Оно позволяет оптимизировать процессы управления данными, улучшить обслуживание клиентов и повысить эффективность работы предприятия в целом.

## Назначение и область применения программного продукта

Разработанное приложение для предприятия по прокату автомобилей в Windows Forms на языке программирования C# предоставляет широкий спектр функций для удобного и эффективного управления всеми аспектами проката автомобилей.

Одной из ключевых функций приложения является управление базами данных клиентов, автомобилей и состояний автомобилей. Работники предприятия могут легко добавлять, редактировать и удалять информацию о клиентах, включая паспортные и контактные данные, ФИО и категорию водительских прав. Также имеется возможность вести учет автомобилей, включая информацию о марке, модели и других характеристиках. Для каждого автомобиля также можно установить его текущий статус, указав, свободен он или занят на аренду.

Приложение также позволяет управлять договорами аренды. Работники предприятия могут добавлять и удалять договоры, указывая данные клиента, автомобиля, даты начала и окончания аренды.

Важным функционалом приложения является возможность просмотра доступных автомобилей в реальном времени. Это позволяет работникам с легкостью находить свободные автомобили для аренды в данный момент, что улучшает их опыт использования услуги проката автомобилей.

Все операции по управлению базами данных и изменению статусов автомобилей в приложении выполняются с помощью интуитивно понятного и простого интерфейса. Работники предприятия могут легко освоить функционал приложения и быстро выполнять необходимые операции, что поможет сократить время на административные задачи и повысить эффективность работы предприятия.

В свете вышесказанного, данное приложение предоставляет всестороннюю поддержку для предприятий по прокату автомобилей, позволяя эффективно управлять базами данных клиентов, автомобилей, состояний автомобилей, договоров аренды и цен на прокат. Оно помогает предприятиям автоматизировать процессы управления, повысить уровень обслуживания клиентов и улучшить результативность бизнеса.

## Постановка задачи

### Определение требований к программной системе

К функциональным требованиям данного приложения относится:

* Вход в систему: Работник должен иметь возможность войти в приложение, используя фиксированный логин и пароль.
* Просмотр базы данных автомобилей, клиентов, состояний автомобилей, договоров и цен на аренду автомобилей: Работник должен иметь возможность просмотреть все базы данных, необходимые для работы предприятия по прокату автомобиля.
* Управление автомобилями: Работник должен иметь возможность добавлять новые автомобили, удалять существующие и изменять их статус.
* Управление клиентами: Работник должен иметь возможность просматривать список клиентов, добавлять, изменять или удалять информацию о клиентах.
* Управление состояниями автомобилей: Работник должен иметь возможность добавлять произведённые технические работы автомобилей и их дату.
* Управление договорами: Работник должен иметь возможность добавлять информацию о договорах (ID клиента, ID автомобиля, дата заключения договора и дата окончания договора), а также удалять её.
* Управление ценами на аренду автомобилей: Работник должен иметь возможность добавлять(обновлять) цену на аренду автомобиля и дату.

К нефункциональным требованиям данного приложения относится:

* Интерфейс пользователя: Приложение должно иметь интуитивно понятный и привлекательный пользовательский интерфейс, разработанный в стиле Windows Forms.
* Производительность: Приложение должно быть отзывчивым и иметь быструю обработку запросов пользователей.
* Совместимость: Приложение должно быть совместимо с операционными системами Windows, поддерживающими Windows Forms.
* Тестирование: Приложение должно проходить тщательное тестирование, чтобы обнаружить и исправить ошибки и недочёты.
* Оптимизация: Приложение должно быть оптимизировано для работы с базой данных, чтобы обеспечить быстрый доступ к информации.

К системным требованиям данного приложения относится:

* Операционная система: Windows 7 и выше.
* Процессор: Intel Core i3 или эквивалентный процессор.
* Оперативная память: минимум 4 ГБ RAM.
* Видеокарта: с поддержкой DirectX 10 и выше.
* Свободное место на жёстком диске: минимум 100 МБ.
* .NET Framework 4.5 и выше.
* Установленная IDE для разработки на C#, например, Visual Studio.

### Выбор и обоснование средств реализации

При выборе средств для реализации приложения для предприятия по прокату автомобилей, учитывались следующих факторы:

* Основные возможности Windows Forms: Windows Forms обеспечивает широкий набор инструментов и элементов управления, которые позволяют создавать интуитивно понятные пользовательский интерфейс. Это включает в себя кнопки, текстовые поля, списки, таблицы и др.
* Язык программирования C#: C# является основным языком программирования для Windows Forms и обеспечивает мощные средства разработки приложений. C# позволяет легко создавать объектно-ориентированный код, поддерживает многопоточность и имеет большую поддержку сообщества разработчиков.
* Гибкость: Windows Forms позволяет создавать не только простые формы, но и сложные интерфейсы, включающие в себя множество элементов управления и функциональности. Можно легко настраивать внешний вид элементов управления, выполнять валидацию ввода и обработку событий.

## Описание структуры ПС

#### Структура входных и выходных данных

Структура входных и выходных данных для приложения для проката автомобилей в Windows Forms на C# может быть следующей:

1. Входные данные:

* Логин и пароль: строки, содержащие фиксированный логин и пароль для входа в систему.
* Поиск: строка, в которую вводится данные для поиска по базе данных.

1. Выходные данные:

* Список свободных автомобилей: база данных со статусов автомобиля «Свободен».
* Данные поиска: строки в базе данных, которые показываются после введения данных в поиск.

#### Описание процедур и функций

1. Процедура аутентификации:

* Проверка введённых пользователем данных на соответствие фиксированным логину и паролю.
* В случае успешной аутентификации открывается главное окно приложения, в противном случае выводится сообщение об ошибке.

1. Функция поиска автомобиля по модели:

* Принимает на вход необходимую модель автомобиля.
* Ищет в базе данных соответствующую строку.
* Выделяет строку с данными соответствующего автомобиля.

1. Функция поиска клиентов по ФИО:

* Принимает на вход ФИО клиента, которого необходимо найти.
* Ищет в базе данных соответствующую строку.
* Выделяет строку с соответствующими данными.

## Проектирование

##### Структура БД

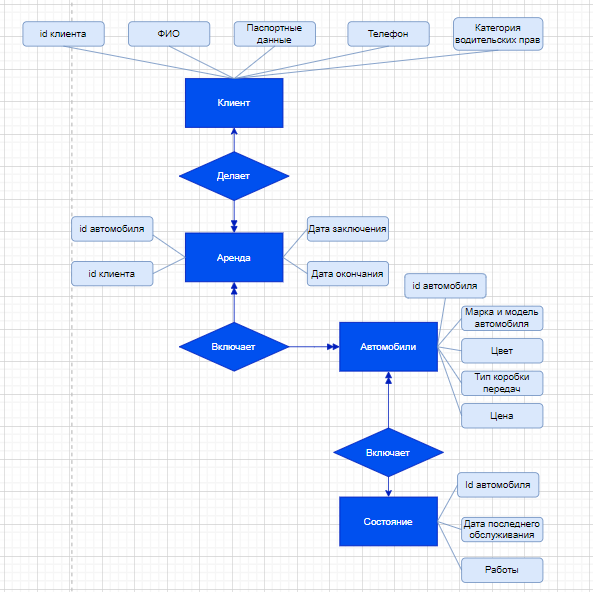


Рисунок 4.1.1 – Концептуальная модель базы данных

Первичным ключом базы данных данного приложения является столбец id клиента в таблице Клиенты. Внешний ключ в таблице Аренда составляют столбцы id клиента, id автомобиля и Дата заключения; в таблице Цена – id автомобиля и Дата. Внутренними ключами базы данных данного приложения является столбец id клиента и id автомобиля в таблице Аренда; id автомобиля в таблице Автомобили; id автомобиля в таблицах Состояние и Цена.

Изображение выглядит как текст, диаграмма, снимок экрана, линия

Автоматически созданное описание

Рисунок 4.1.2 – Логическая модель базы данных

##### Описание решений по архитектуре

Для приложения для проката автомобиля в Windows Forms на C# была выбрана архитектура, основанная на трехслойной архитектуре Model-View-Presenter (MVP). Эта архитектура позволяет разделить логику приложения, представление данных и управление пользовательским интерфейсом на отдельные слои, что упрощает поддержку и расширение приложения.

1. Диаграмма вариантов использования:

На диаграмме вариантов использования представлены основные функциональные возможности приложения:

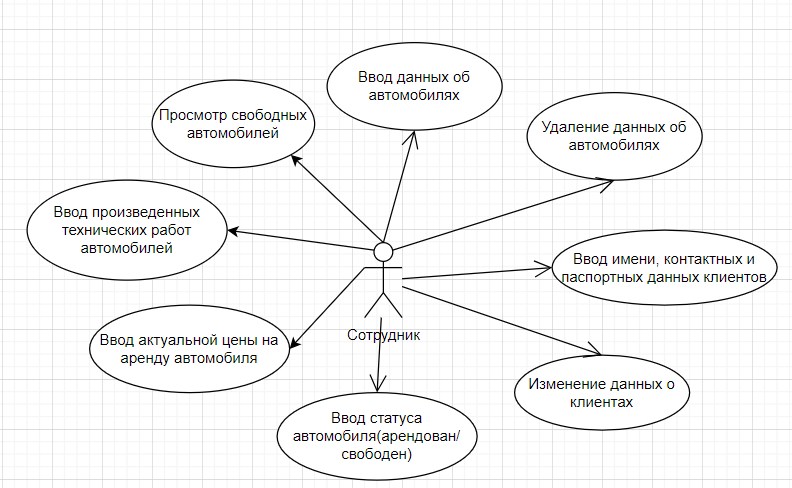


Рисунок 4.2.1 – Диаграмма вариантов использования

1. Диаграмма классов приложения:

Диаграмма классов приложения показывает связи между классами и их атрибутами и методами. В данном приложении выделены следующие классы:

* Класс «Клиент» - содержит информацию о клиентах, имеет связь с классом «Аренда».
* Класс «Автомобили» - хранит информацию об автомобилях, имеет связь с классами «Аренда», «Состояние», «Цена».
* Класс «Аренда» - содержит информацию о договорах аренды, имеет связь с классами "Клиент" и "Автомобили".
* Класс «Состояние» - содержит информацию о технических работах автомобилей, имеет связь с классом «Автомобили».
* Класс «Цена» - содержит информацию о стоимости проката автомобиля, имеет связь с классов «Автомобили».

Изображение выглядит как текст, диаграмма, Параллельный, линия

Автоматически созданное описание

Рисунок 4.2.2 – Диаграмма классов

1. Диаграмма состояния:

Диаграмма состояния показывает возможные состояния объектов в приложении и переходы между ними.

* Состояние "Ожидание получения статуса автомобиля" - когда оператор отправляет запрос, свободен ли необходимый автомобиль.
* Состояние "Получение статуса автомобиля" – когда оператор просмотрел статус автомобиля.
* Состояние "Подтверждение аренды" - когда автомобиль не арендован и доступен для проката.
* Состояние "Отклонение аренды" - когда автомобиль арендован и недоступен для проката.

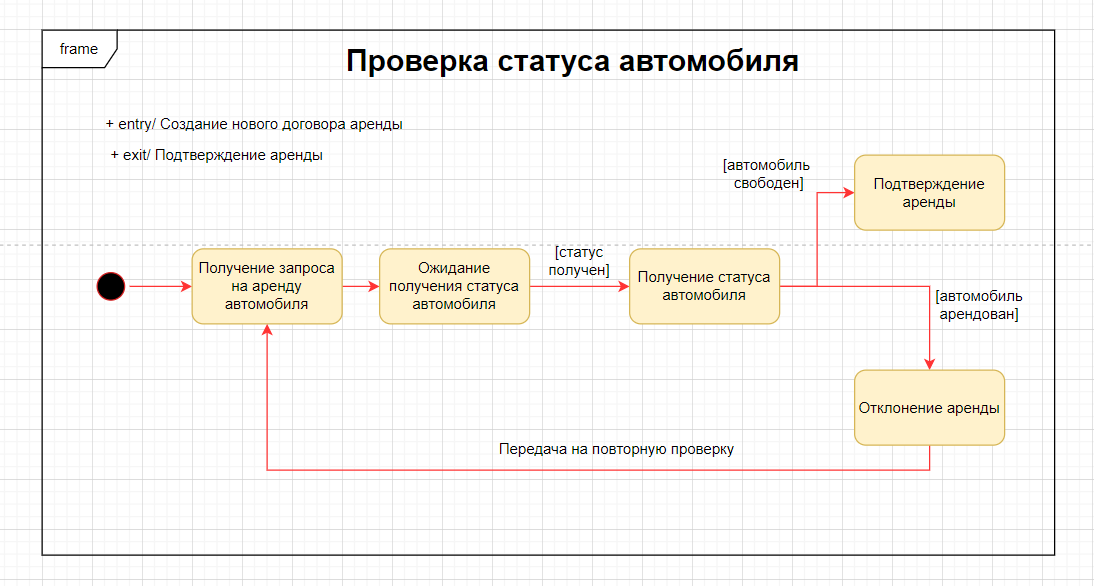


Рисунок 4.2.3 – Диаграмма состояний

1. Диаграмма деятельности:

Диаграмма деятельности показывает последовательность действий, которые выполняются в приложении при выполнении определённой задачи. В данном приложении выделены следующие действия:

* Действие «Запрашивает аренду требуемого автомобиля» - процесс аренды автомобиля, который запрос на необходимый для аренды автомобиль.
* Действие «Проверка статуса автомобиля» - процесс аренды автомобиля, при котором просматривается статус автомобиля для дальнейших действий.
* Действие «Ввод данных клиента в БД» - процесс добавления нового клиента в базу данных.
* Действие «Оформление договора аренды» – процесс аренды автомобиля, который включает выбор автомобиля, заполнение данных о клиенте и подтверждение аренды.
* Действие «Ожидание оплаты» - процесс аренды автомобиля, при котором после оформления договора, клиент оплачивает аренду.
* Действие «Выдача автомобиля клиенту» - процесс аренды автомобиля, при котором после оформления договора и оплаты аренды клиенту выдаётся автомобиль.
* Действие «Связываемся с клиентом» - процесс аренды автомобиля, при котором оператор связывается с клиентом, если последний вовремя не вернул автомобиль.
* Действие «Меняем статус автомобиля» - процесс аренды автомобиля, при котором мы меняем статус автомобиля со «свободен» на «арендован» и наоборот.

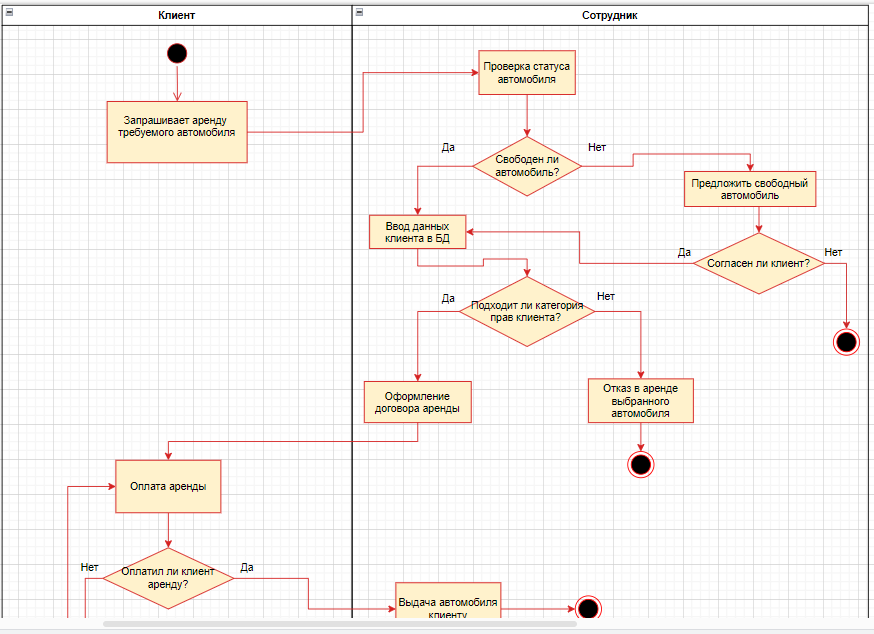


Рисунок 4.2.4 – Диаграмма деятельности

Изображение выглядит как текст, диаграмма, снимок экрана, линия

Автоматически созданное описание

Рисунок 4. 2. 5 – Диаграмма деятельности. Продолжение

1. Диаграмма последовательности:

Диаграмма последовательности показывает взаимодействие между объектами в приложении при выполнении определённой задачи. В данном приложении выделены следующие взаимодействия:

* Взаимодействие «Авторизация» - между объектами «Работник» и приложением.
* Взаимодействие «Ввод данных об автомобилях» - между объектами "Автомобиль", и базой данных.
* Взаимодействие «Удаление автомобиля» - между объектом "Автомобиль" и базой данных.
* Взаимодействие «Ввод имени, контактных и паспортных данных клиентов» - между объектом "Клиенты" и базой данных.
* Взаимодействие «Изменение данных клиентов» - между объектом "Клиент" и базой данных.
* Взаимодействие «Ввод статуса автомобиля(арендован/свободен)» - между объектом "Автомобиль" и базой данных.

**Изображение выглядит как текст, диаграмма, снимок экрана, Параллельный

Автоматически созданное описание**

Рисунок 4.2.6 – Диаграмма последовательности «Авторизация»

Изображение выглядит как текст, снимок экрана, диаграмма, число

Автоматически созданное описание

Рисунок 4.2.7 - Диаграмма последовательности «Ввод данных об автомобилях»

Изображение выглядит как текст, снимок экрана, число, Параллельный

Автоматически созданное описание

Рисунок 4.2.8 - Диаграмма последовательности «Удаление данных об автомобилях»

Изображение выглядит как текст, снимок экрана, число, диаграмма

Автоматически созданное описание

Рисунок 4.2.9 - Диаграмма последовательности «Ввод имени, контактных и паспортных данных клиентов»

Изображение выглядит как текст, диаграмма, Параллельный, число

Автоматически созданное описание

Рисунок 4.2.10 - Диаграмма последовательности «Изменение данных клиентов»

Изображение выглядит как текст, снимок экрана, число, диаграмма

Автоматически созданное описание

Рисунок 4.2.11 - Диаграмма последовательности «Ввод статуса автомобиля(арендован/свободен)»

##### Проектирование интерфейса

Для проектирования интерфейса приложения для проката автомобилей в Windows Forms на C# были использованы следующие элементы управления:

* Форма «Авторизация» – содержит поля для ввода логина и пароля, а также кнопку входа, которая проверяет введённые данные.
* Форма «Главное меню» - содержит кнопки для доступа к основным функциям приложения: «Клиенты», «Автомобили», «Состояние автомобилей», «Договоры», «Цены».
* Форма «Клиенты» - содержит элементы для просмотра и работы с базой данных клиентов.
* Форма «Автомобили» - содержит элементы для просмотра и работы с базой данных автомобилей, а также содержит кнопку для просмотра свободных автомобилей на данный момент.
* Форма «Состояние автомобиля» - содержит элементы для просмотра и работы с базой данных состояний автомобиля.
* Форма «Договоры» - содержит элементы для просмотра и работы с базой данных договоров.
* Форма «Цены» - содержит элементы для просмотра и работы с базой данных цен.

Все формы имеют единый стиль оформления и цветовую гамму, что обеспечивает единообразный и легко узнаваемый интерфейс приложения. Также были использованы элементы визуального оформления, такие как иконки и изображения, для повышения привлекательности и удобства использования приложения.

## Описание интерфейса программного продукта

Интерфейс программного продукта представляет собой набор форм, позволяющих пользователю управлять процессом проката автомобилей. Все формы имеют простой и интуитивно понятный дизайн, что делает использование приложения удобным для любого пользователя.

Форма авторизации содержит поля для ввода логина и пароля, а также кнопку входа, которая проверяет введённые данные.

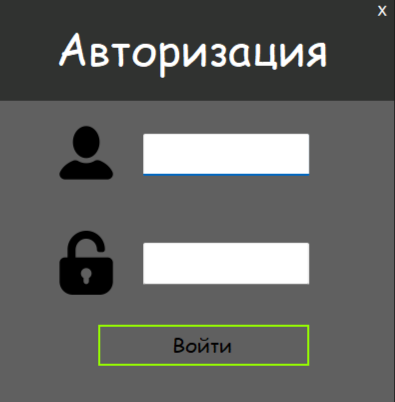


Рисунок 5.1 – Форма «Авторизация»

Форма "Главное меню" - содержит кнопки для доступа к основным функциям приложения: «Клиенты», «Автомобили», «Состояние автомобилей», «Договоры», «Цены».

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание

Рисунок 5.2 – Форма «Главная»

Форма "Клиенты" содержит список всех клиентов, которые когда-либо брали автомобиль в прокат.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок 5.3 – Форма «Клиенты»

Форма «Модификация данных клиентов» содержит поля и кнопки для работы с базой данных клиентов. Здесь можно добавлять новых клиентов, редактировать их данные или удалять из базы.

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание

Рисунок 5.4 – Форма «Модификация данных клиентов»

Форма "Автомобили" представляет собой таблицу с информацией обо всех доступных для проката автомобилях. Также в этой форме можно осуществлять поиск по модели.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок 5.5 – Форма «Автомобили»

Форма «Модификация данных автомобилей» содержит поля и кнопки для работы с базой данных автомобилей. Здесь пользователь может добавлять новые автомобили или удалять уже имеющиеся.

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание

Рисунок 5.6 - Форма «Модификация данных автомобилей»

Форма «Свободные автомобили» содержит список всех свободных на данный момент автомобилей.

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Автоматически созданное описание

Рисунок 5.7 – Форма «Свободные автомобили»

Форма «Состояние автомобилей» содержит список технических работ, проведённых с автомобилями.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок 5.8 – Форма «Состояние автомобилей»

Форма «Модификация данных состояний автомобилей» содержит поля и кнопки для работы с базой данных состояний автомобилей. Здесь пользователь может добавлять новые технические работы и их дату.

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рисунок 5.9 – Форма «Модификация данных состояний автомобилей»

Форма «Цены» содержит список цен на аренду автомобилей. Там есть как актуальные цены, так и устаревшие для просмотра изменений.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок 5.10 – Форма «Цены»

Форма «Модификация данных цен» содержит поля и кнопки для работы с базой данных цен. Здесь пользователь может добавить новую цену на аренду автомобиля, тем самым актуализировать стоимость.

Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Автоматически созданное описание

Рисунок 5.11 – Форма «Модификация данных цен»

Форма «Договоры» содержит список всех договоров аренды, которые были у предприятия.

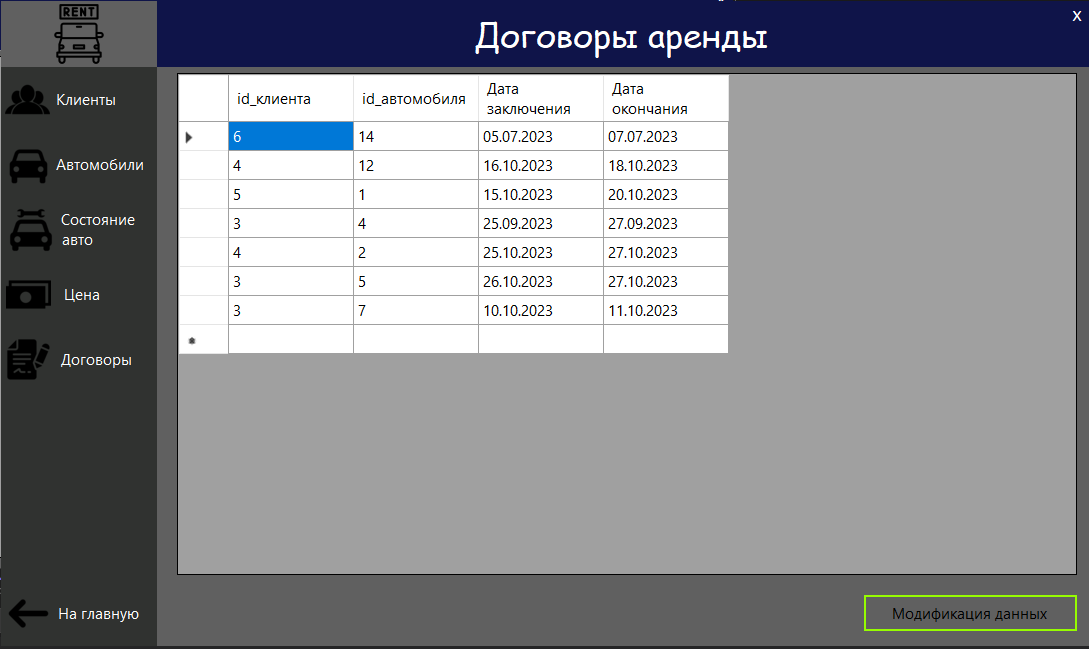


Рисунок 5.12 – Форма «Договоры аренды»

Форма «Модификация данных договоров» содержит поля и кнопки для работы с базой данных договоров. Здесь пользователь может добавить новый договор или удалить уже имеющийся.

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание

Рисунок 5.13 – Форма «Модификация данных договоров»

При проектировании интерфейса были приняты следующие решения для повышения юзабилити:

* Использование простых и понятных названий для форм и элементов управления.
* Размещение кнопок для основных действий на главном окне и в каждой форме, чтобы пользователь мог быстро получить доступ к нужной ему функциональности.
* Использование цветовой схемы, соответствующей стандартам операционной системы Windows, для создания единого стиля приложения.
* Размещение информации в виде таблиц и списков, что облегчает восприятие и поиск необходимых данных.
* Реализация возможности поиска данных для удобства работы с большим объёмом информации.

Таким образом, интерфейс программного продукта является интуитивно понятным и удобным для использования, что позволяет пользователям быстро освоить работу с приложением, уменьшить время обработки информации и повысить эффективность своей деятельности.

## Шаблоны проектирования

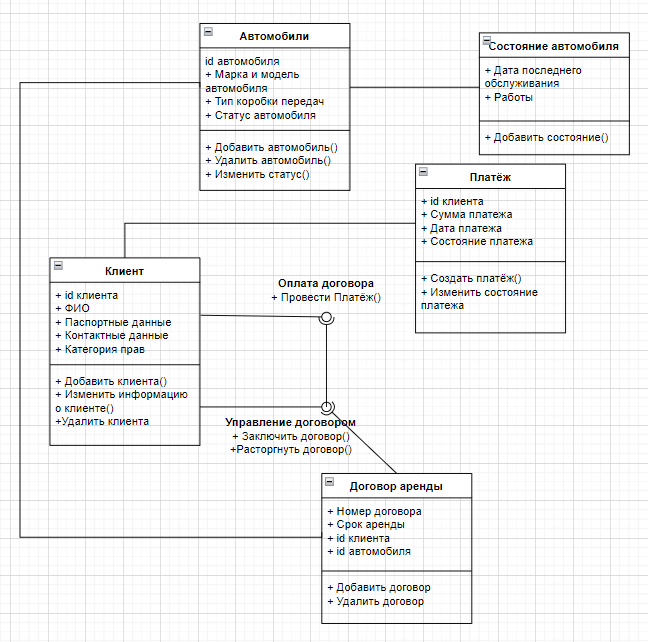


Рисунок 6 – Диаграмма классов

* Класс «Состояние автомобиля» является дочерним классом класса «Автомобили». В своих методах имеет возможность добавить дату обслуживания и проведённую техническую работу.
* Класс «Автомобили» является родительским классом класса «Состояние автомобиля», имеет ассоциативную связь с классом «Договоры аренды» и передаёт туда id автомобиля. Имеет 3 метода: добавить автомобиль, удалить автомобиль и изменить статус автомобиля.
* Класс «Договор аренды» является связывающим классом в приложении. Этот класс содержит в себе компоненты id автомобиля из класса «Автомобили» и id клиента из класса «Клиенты». Имеет 2 метода: добавить договор и удалить договор.
* Класс «Клиенты» связан с классом «Договор аренды» и «Платёж» и передаёт в эти классы компонент id клиента. Имеет 3 метода: добавить информацию о клиенте, изменить информацию и удалить клиента.
* Класс «Платёж» связан с классами «Клиент» и «Договор аренды». Имеет 2 метода: создать платёж и изменить состояние платежа.

## Тестирование программного продукта

Для уверенности в том, что программа будет работать так, как было задумано, было проведено тестирование.

Тестирование – процесс, подтверждающий правильность программы и демонстрирующий, что ошибок в программе нет. Тестирование проводится с намерением найти ошибки, если они имеются.

ПО написано в программе Visual Studio 2022.

Список общих проверок на правильность выполнения:

* Переходы между формами работают корректно.
* Вывод базы данных на формы работает корректно.

Таблица 7.1 – Результат тестирования формы «Авторизация»

|  |  |  |
| --- | --- | --- |
| **Функция** | **Ожидаемый результат** | **Фактический результат** |
| Кнопка «Войти» с незаполненным textBoxLogin. | Появление окна с сообщением «Логин или пароль неверный, попробуйте ещё раз!». | Появление окна с сообщением. |
| Кнопка «Войти» с незаполненным textBoxPassword. | Появление окна с сообщением «Логин или пароль неверный, попробуйте ещё раз!». | Появление окна с сообщением. |
| Кнопка «Войти» с неправильно заполненным textBoxLogin и/или textBoxPassword. | Появление окна с сообщением «Логин или пароль неверный, попробуйте ещё раз!». | Появление окна с сообщением. |
| Кнопка «Войти» с правильно заполненным textBoxLogin(admin) и textBoxPassword(operator). | Переход на главную форму для выбора действий. | Переход на главную форму. |

Таблица 7.2 – Тестирование поиска

|  |  |  |
| --- | --- | --- |
| **Функция** | **Ожидаемый результат** | **Фактический результат** |
| Кнопка «Найти» при отсутствии введённых данных в БД. | Появление окна с сообщением «Данные не найдены!». | Появление окна с сообщением. |
| Кнопка «Найти» при присутствии данных в БД. | Выделение строк(и) с подходящими данными. | Выделение строки. |

Таблица 7.3 – Тестирование изменения данных в БД

|  |  |  |
| --- | --- | --- |
| **Функция** | **Ожидаемый результат** | **Фактический результат** |
| Кнопка «Сохранить» при не всех введённых данных в textBox и выбран RadioButtonAdd. | Появление окна с сообщением «Данные успешно сохранены и обновлены в базе данных» и внесение всех введённых данных в БД. | Появление окна с сообщением и внесение всех введённых данных в БД, строка с данными, где не было ничего введено остаётся пустой. |
| Кнопка «Сохранить» при не всех введённых данных в textBox и выбран RadioButtonUpdate. | Появление окна с сообщением «Данные успешно сохранены и обновлены в базе данных» и обновление всех введённых данных в БД. | Появление окна с сообщением и внесение всех введённых данных в БД, строка с данными, где не было ничего введено остаётся не изменённой. |
| Кнопка «Сохранить» при введённых данных в textBox и выбран RadioButtonDelete. | Появление окна с сообщением «Данные успешно сохранены и обновлены в базе данных» и удаление выбранной строки. | Появление окна с сообщением и удаление выбранной строки. |
| Кнопка «Сохранить» при не введённых данных в textBox и выбран RadioButtonDelete. | Появление окна с сообщением «Данные успешно сохранены и обновлены в базе данных», но никакие данные не удаляются. | Появление окна с сообщением, но никакие данные не удаляются. |
| Кнопка «Сохранить» при всех введённых данных в textBox и выбран RadioButtonAdd. | Появление окна с сообщением «Данные успешно сохранены и обновлены в базе данных» и внесение всех введённых данных в БД. | Появление окна с сообщением и внесение всех введённых данных в БД. |
| Кнопка «Сохранить» при всех введённых данных в textBox и выбран RadioButtonUpdate. | Появление окна с сообщением «Данные успешно сохранены и обновлены в базе данных» и изменение введённых данных в БД. | Появление окна с сообщением и внесение всех введённых данных в БД. |
| Кнопка «Сохранить» при всех введённых данных в textBox и не выбран RadioButton. | Ничего не произойдёт. | Ничего не происходит. |

Таблица 7.4 – Тестирование формы «Автомобили»

|  |  |  |
| --- | --- | --- |
| **Функция** | **Ожидаемый результат** | **Фактический результат** |
| Кнопка «Модификация данных». | Переход на форму «Модификация данных автомобилей» для редактирования данных автомобилей. | Переход на форму. |
| Кнопка «Свободные автомобили». | Переход на форму «Свободные автомобили». | Переход на форму. |
| Кнопка «Клиенты» в боковом меню. | Переход на форму «Клиенты». | Переход на форму. |
| Кнопка «Автомобили» в боковом меню. | Ничего не произойдёт. | Ничего не происходит. |
| Кнопка «Состояние авто» в боковом меню. | Переход на форму «Состояние авто». | Переход на форму. |
| Кнопка «Цена» в боковом меню. | Переход на форму «Цена». | Переход на форму. |
| Кнопка «Договоры» в боковом меню. | Переход на форму «Договоры». | Переход на форму. |

# Заключение

Создание интерфейса приложения является одним из важнейших этапов в разработке любого программного продукта. Именно интерфейс обеспечивает взаимодействие пользователя с приложением и влияет на удобство и эффективность его использования. В данной работе было изучено создание интерфейса приложения с использованием элементов управления Windows Forms, которые позволяют разработчику создавать графический интерфейс для приложения, не прибегая к написанию сложного кода.

Основными элементами управления Windows Forms являются кнопки, текстовые поля и другие элементы, которые можно расположить на форме приложения и настроить их свойства для достижения нужного внешнего вида и функциональности. В результате было разработано приложение, позволяющее работать с базой данных предприятия, в которой есть информацию о клиентах, автомобилях, состояниях автомобилей, договорах аренды и стоимости на аренду автомобилей. Добавлять, изменять и удалять информацию, которая содержится в базе данных, а также реализован просмотр свободных автомобилей на данный момент.

При разработке интерфейса приложения были учтены основные принципы оформления, такие как единообразие, простота, наглядность и удобство использования. Также были использованы элементы визуального оформления, такие как цвета, шрифты и изображения, для создания привлекательного и удобочитаемого интерфейса.

В результате было разработано удобное и привлекательное приложение, которое может быть использовано в качестве примера для создания подобных приложений. Кроме того, оно может быть доработано и усовершенствовано в дальнейшем для добавления новых функций и улучшения внешнего вида. Таким образом, изучение создания интерфейса приложения с использованием элементов управления Windows Forms является важным шагом для развития навыков программирования и создания качественных программных продуктов.

# Список использованной литературы

1. О. Н. Евсеева, А. Б. Шамшев – Работа с базами данных на языке C# (2009)
2. Microsoft Learn [Электронный ресурс] – Режим доступа: <https://learn.microsoft.com/ru-ru/dotnet/csharp/>
3. Учебник по C# [Электронный ресурс] – Режим доступа: <https://csharp.webdelphi.ru/uchebnik-po-c/>
4. Справочник по C# [Электронный ресурс] – Режим доступа: <https://www.cyberforum.ru/>

# Приложение А

//Форма "Авторизация"

namespace Прокат

{

public partial class authorization : Form

{

public authorization()

{

InitializeComponent();

}

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

{

Application.Exit();

}

private void labelClose\_MouseEnter(object sender, EventArgs e)

{

labelClose.ForeColor = Color.Red;

}

private void labelClose\_MouseLeave(object sender, EventArgs e) //изменение цвета кнопки "x" при наведении курсора

{

labelClose.ForeColor = Color.White;

}

Point lastPoint;

private void panel2\_MouseMove(object sender, MouseEventArgs e) //возможность менять положение окна на экране

{

if (e.Button == MouseButtons.Left)

{

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

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

}

}

private void panel2\_MouseDown(object sender, MouseEventArgs e) //возможность менять положение окна на экране

{

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

}

private void buttonEnter\_Click(object sender, EventArgs e) //проверка логина и пароля

{

string login = textBoxLogin.Text;

string password = textBoxPassword.Text;

if (login == "admin" && password == "operator")

{

Main formMain = new Main();

formMain.Show();

this.Hide();

}

else

{

textBoxPassword.Text = string.Empty;

MessageBox.Show("Логин или пароль неверный, попробуйте ещё раз!", "Ошибка");

}

}

}

//Форма "Главная"

namespace Прокат

{

public partial class Main : Form

{

public Main()

{

InitializeComponent();

}

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

{

Application.Exit();

}

private void labelClose\_MouseEnter(object sender, EventArgs e)

{

labelClose.ForeColor = Color.Red;

}

private void labelClose\_MouseLeave(object sender, EventArgs e)

{

labelClose.ForeColor = Color.White;

}

Point lastPoint;

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

{

if (e.Button == MouseButtons.Left)

{

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

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

}

}

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

{

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

}

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

{

Clients formClients = new Clients();

formClients.Show();

this.Hide();

}

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

{

Cars formCars = new Cars();

formCars.Show();

this.Hide();

}

private void buttonConditionCars\_Click\_1(object sender, EventArgs e)

{

Condition formCondition = new Condition();

formCondition.Show();

this.Hide();

}

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

{

Price formPrice = new Price();

formPrice.Show();

this.Hide();

}

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

{

Agreement formAgreement = new Agreement();

formAgreement.Show();

this.Hide();

}

}

}

//Форма "Клиенты"

namespace Прокат

{

public partial class Clients : Form

{

public static string str = @"Provider=Microsoft.Ace.OLEDB.12.0;Data Source='D:\Study\3 курс\Практика ТРПО\Прокат\БД.accdb';";

string comm = "Select \* from Клиент";

public Clients() //Подключение БД

{

InitializeComponent();

OleDbConnection con = new OleDbConnection(str);

con.Open();

OleDbDataAdapter ad = new OleDbDataAdapter(comm, con);

DataTable ds = new DataTable();

ad.Fill(ds);

dataGridView1.AutoGenerateColumns = true;

dataGridView1.DataSource = ds;

con.Close();

}

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

{

Application.Exit();

}

private void labelClose\_MouseEnter(object sender, EventArgs e) //изменение цвета кнопки "x" при наведении курсора

{

labelClose.ForeColor = Color.Red;

}

private void labelClose\_MouseLeave(object sender, EventArgs e) //изменение цвета кнопки "x" при наведении курсора

{

labelClose.ForeColor = Color.White;

}

Point lastPoint;

private void panelClients\_MouseMove(object sender, MouseEventArgs e) //возможность менять положение окна на экране

{

if (e.Button == MouseButtons.Left)

{

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

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

}

}

private void panelClients\_MouseDown(object sender, MouseEventArgs e) //возможность менять положение окна на экране

{

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

}

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

{

Clients formClients = new Clients();

formClients.Show();

this.Hide();

}

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

{

Cars formCars = new Cars();

formCars.Show();

this.Hide();

}

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

{

Condition formCondition = new Condition();

formCondition.Show();

this.Hide();

}

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

{

Price formPrice = new Price();

formPrice.Show();

this.Hide();

}

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

{

Agreement formAgreement = new Agreement();

formAgreement.Show();

this.Hide();

}

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

{

Main formMain = new Main();

formMain.Show();

this.Hide();

}

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

{

Data\_Modification\_Client formModificationClient = new Data\_Modification\_Client();

formModificationClient.Show();

}

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

{

string searchText = textBoxSearch.Text.ToLower(); // Получение текста из TextBox для поиска и приведение его к нижнему регистру

try

{

foreach (DataGridViewRow row in dataGridView1.Rows) // Перебор всех строк в DataGridView1

{

string clientFIO = row.Cells["ФИО"].Value.ToString().ToLower(); // Получение ФИО клиента из соответствующей ячейки в строке и приведение его к нижнему регистру

if (clientFIO.Contains(searchText)) // Проверка наличия введенного текста в ФИО клиента

{

row.Selected = true; // Выделение строки, если найдено совпадение

}

else

{

row.Selected = false; // Снятие выделения, если совпадение не найдено

MessageBox.Show("Данные не найдены!");

}

}

}

catch (Exception) { }

}

}

}

//Форма "Автомомобили"

namespace Прокат

{

public partial class Cars : Form

{

public static string str = @"Provider=Microsoft.Ace.OLEDB.12.0;Data Source='D:\Study\3 курс\Практика ТРПО\Прокат\БД.accdb';";

string comm = "Select \* from Автомобили";

public Cars()

{

InitializeComponent();

OleDbConnection con = new OleDbConnection(str);

con.Open();

OleDbDataAdapter ad = new OleDbDataAdapter(comm, con);

DataTable ds = new DataTable();

ad.Fill(ds);

dataGridView1.AutoGenerateColumns = true;

dataGridView1.DataSource = ds;

con.Close();

}

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

{

Application.Exit();

}

private void labelClose\_MouseEnter(object sender, EventArgs e)

{

labelClose.ForeColor = Color.Red;

}

private void labelClose\_MouseLeave(object sender, EventArgs e)

{

labelClose.ForeColor = Color.White;

}

Point lastPoint;

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

{

if (e.Button == MouseButtons.Left)

{

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

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

}

}

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

{

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

}

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

{

Clients formClients = new Clients();

formClients.Show();

this.Hide();

}

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

{

Cars formCars = new Cars();

formCars.Show();

this.Hide();

}

private void buttonCondition\_Click\_1(object sender, EventArgs e)

{

Condition formCondition = new Condition();

formCondition.Show();

this.Hide();

}

private void buttonPrice\_Click\_1(object sender, EventArgs e)

{

Price formPrice = new Price();

formPrice.Show();

this.Hide();

}

private void buttonAgreement\_Click\_1(object sender, EventArgs e)

{

Agreement formAgreement = new Agreement();

formAgreement.Show();

this.Hide();

}

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

{

Main formMain = new Main();

formMain.Show();

this.Hide();

}

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

{

Data\_modification\_Car formModificationCar = new Data\_modification\_Car();

formModificationCar.Show();

}

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

{

try

{

string searchText = textBoxSearch.Text.ToLower(); // Получение текста из TextBox для поиска и приведение его к нижнему регистру

foreach (DataGridViewRow row in dataGridView1.Rows) // Перебор всех строк в DataGridView1

{

string ModelCar = row.Cells["Модель автомобиля"].Value.ToString().ToLower(); // Получение модели автомобиля из соответствующей ячейки в строке и приведение его к нижнему регистру

if (ModelCar.Contains(searchText)) // Проверка наличия введенного текста в ФИО клиента

{

row.Selected = true; // Выделение строки, если найдено совпадение

}

else

{

row.Selected = false; // Снятие выделения, если совпадение не найдено

MessageBox.Show("Данные не найдены!");

}

}

}

catch (Exception) { }

}

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

{

Available\_cars formAvailableCars = new Available\_cars();

formAvailableCars.Show();

}

}

}

//Форма "Договоры"

namespace Прокат

{

public partial class Agreement : Form

{

public static string str = @"Provider=Microsoft.Ace.OLEDB.12.0;Data Source='D:\Study\3 курс\Практика ТРПО\Прокат\БД.accdb';";

string comm = "Select \* from Аренда";

public Agreement()

{

InitializeComponent();

OleDbConnection con = new OleDbConnection(str);

con.Open();

OleDbDataAdapter ad = new OleDbDataAdapter(comm, con);

DataTable ds = new DataTable();

ad.Fill(ds);

dataGridView1.AutoGenerateColumns = true;

dataGridView1.DataSource = ds;

con.Close();

}

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

{

Application.Exit();

}

private void labelClose\_MouseEnter(object sender, EventArgs e)

{

labelClose.ForeColor = Color.Red;

}

private void labelClose\_MouseLeave(object sender, EventArgs e)

{

labelClose.ForeColor = Color.White;

}

Point lastPoint;

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

{

if (e.Button == MouseButtons.Left)

{

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

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

}

}

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

{

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

}

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

{

Clients formClients = new Clients();

formClients.Show();

this.Hide();

}

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

{

Cars formCars = new Cars();

formCars.Show();

this.Hide();

}

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

{

Condition formCondition = new Condition();

formCondition.Show();

this.Hide();

}

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

{

Price formPrice = new Price();

formPrice.Show();

this.Hide();

}

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

{

Agreement formAgreement = new Agreement();

formAgreement.Show();

this.Hide();

}

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

{

Main formMain = new Main();

formMain.Show();

this.Hide();

}

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

{

Data\_modification\_Agreement formModificationAgreement = new Data\_modification\_Agreement();

formModificationAgreement.Show();

}

}

}

//Форма "Цены"

namespace Прокат

{

public partial class Price : Form

{

public static string str = @"Provider=Microsoft.Ace.OLEDB.12.0;Data Source='D:\Study\3 курс\Практика ТРПО\Прокат\БД.accdb';";

string comm = "Select \* from Цена order by id\_автомобиля,Дата";

public Price()

{

InitializeComponent();

OleDbConnection con = new OleDbConnection(str);

con.Open();

OleDbDataAdapter ad = new OleDbDataAdapter(comm, con);

DataTable ds = new DataTable();

ad.Fill(ds);

dataGridView1.AutoGenerateColumns = true;

dataGridView1.DataSource = ds;

con.Close();

}

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

{

Application.Exit();

}

private void labelClose\_MouseEnter(object sender, EventArgs e)

{

labelClose.ForeColor = Color.Red;

}

private void labelClose\_MouseLeave(object sender, EventArgs e)

{

labelClose.ForeColor = Color.White;

}

Point lastPoint;

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

{

if (e.Button == MouseButtons.Left)

{

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

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

}

}

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

{

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

}

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

{

Clients formClients = new Clients();

formClients.Show();

this.Hide();

}

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

{

Cars formCars = new Cars();

formCars.Show();

this.Hide();

}

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

{

Condition formCondition = new Condition();

formCondition.Show();

this.Hide();

}

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

{

Price formPrice = new Price();

formPrice.Show();

this.Hide();

}

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

{

Agreement formAgreement = new Agreement();

formAgreement.Show();

this.Hide();

}

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

{

Main formMain = new Main();

formMain.Show();

this.Hide();

}

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

{

Data\_modification\_Price formModificationPrice = new Data\_modification\_Price();

formModificationPrice.Show();

}

}

}

//Форма "Свободные автомобили"

namespace Прокат

{

public partial class Available\_cars : Form

{

public static string str = @"Provider=Microsoft.Ace.OLEDB.12.0;Data Source='D:\Study\3 курс\Практика ТРПО\Прокат\БД.accdb';";

string comm = "SELECT \* FROM Автомобили WHERE [Статус] = 'Свободен'";

public Available\_cars()

{

InitializeComponent();

}

Point lastPoint;

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

{

if (e.Button == MouseButtons.Left)

{

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

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

}

}

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

{

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

}

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

{

this.Close();

}

private void Available\_cars\_Load(object sender, EventArgs e)

{

OleDbConnection con = new OleDbConnection(str);

con.Open();

OleDbDataAdapter ad = new OleDbDataAdapter(comm, con);

DataTable ds = new DataTable();

ad.Fill(ds);

dataGridView2.AutoGenerateColumns = true;

dataGridView2.DataSource = ds;

con.Close();

}

}

}

//Форма "Модификация данных клиентов"

namespace Прокат

{

public partial class Data\_Modification\_Client : Form

{

public static string str = @"Provider=Microsoft.Ace.OLEDB.12.0;Data Source='D:\Study\3 курс\Практика ТРПО\Прокат\БД.accdb';";

//string comm = "Select \* from Клиент";

public Data\_Modification\_Client()

{

InitializeComponent();

}

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

{

this.Close();

}

Point lastPoint;

private void panel1\_MouseMove(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

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

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

}

}

private void panel1\_MouseDown(object sender, MouseEventArgs e)

{

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

}

private void radioButtonAdd\_CheckedChanged(object sender, EventArgs e)

{

textBoxId.Visible = true;

textBoxFIO.Visible = true;

textBoxPasport.Visible = true;

textBoxTelephone.Visible = true;

textBoxDriver.Visible = true;

labelId.Visible = true;

labelFIO.Visible = true;

labelPasport.Visible = true;

labelTelephone.Visible = true;

labelDriver.Visible = true;

}

private void radioButtonUpdate\_CheckedChanged(object sender, EventArgs e)

{

textBoxId.Visible = true;

textBoxFIO.Visible = true;

textBoxPasport.Visible = true;

textBoxTelephone.Visible = true;

textBoxDriver.Visible = true;

labelId.Visible = true;

labelFIO.Visible = true;

labelPasport.Visible = true;

labelTelephone.Visible = true;

labelDriver.Visible = true;

}

private void radioButtonDelete\_CheckedChanged\_1(object sender, EventArgs e)

{

textBoxId.Visible = true;

textBoxFIO.Visible = false;

textBoxPasport.Visible = false;

textBoxTelephone.Visible = false;

textBoxDriver.Visible = false;

labelId.Visible = true;

labelFIO.Visible = false;

labelPasport.Visible = false;

labelTelephone.Visible = false;

labelDriver.Visible = false;

}

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

{

OleDbConnection con = new OleDbConnection(str);

con.Open();

if (radioButtonAdd.Checked)

{

try

{

string request = "INSERT INTO Клиент ([id\_клиента], [ФИО], [Паспортные данные], [Телефон], [Категория водительских прав]) VALUES (@Id, @FIO, @Pasport, @Telephone, @Driver)";

OleDbCommand command = new OleDbCommand(request, con);

command.Parameters.AddWithValue("@Id", textBoxId.Text);

command.Parameters.AddWithValue("@FIO", textBoxFIO.Text);

command.Parameters.AddWithValue("@Psport", textBoxPasport.Text);

command.Parameters.AddWithValue("@Telephone", textBoxTelephone.Text);

command.Parameters.AddWithValue("@Driver", textBoxDriver.Text);

command.ExecuteNonQuery();

MessageBox.Show("Данные успешно сохранены и обновлены в базе данных.", "Сообщение");

}

catch (Exception ex)

{

MessageBox.Show("Ошибка сохранения данных: " + ex.Message);

}

}

else if (radioButtonUpdate.Checked)

{

try

{

string request = "UPDATE Клиент SET [ФИО] = @FIO, [Паспортные данные] = @Pasport, [Телефон] = @Telephone, [Категория водительских прав] = @Driver WHERE [id\_клиента] = @Id";

OleDbCommand command = new OleDbCommand(request, con);

command.Parameters.AddWithValue("@FIO", textBoxFIO.Text);

command.Parameters.AddWithValue("@Pasport", textBoxPasport.Text);

command.Parameters.AddWithValue("@Telephone", textBoxTelephone.Text);

command.Parameters.AddWithValue("@Driver", textBoxDriver.Text);

command.Parameters.AddWithValue("@Id", textBoxId.Text);

command.ExecuteNonQuery();

MessageBox.Show("Данные успешно сохранены и обновлены в базе данных.", "Сообщение");

}

catch (Exception ex)

{

MessageBox.Show("Ошибка сохранения данных: " + ex.Message);

}

}

else if (radioButtonDelete.Checked)

{

try

{

string request = "DELETE FROM Клиент WHERE [id\_клиента] = @Id";

OleDbCommand command = new OleDbCommand(request, con);

command.Parameters.AddWithValue("@Id", textBoxId.Text);

command.ExecuteNonQuery();

MessageBox.Show("Данные успешно сохранены и обновлены в базе данных.", "Сообщение");

}

catch (Exception ex)

{

MessageBox.Show("Ошибка сохранения данных: " + ex.Message);

}

}

con.Close();

}

}

}

//Форма "Модификация данных автомобилей"

namespace Прокат

{

public partial class Data\_modification\_Car : Form

{

public static string str = @"Provider=Microsoft.Ace.OLEDB.12.0;Data Source='D:\Study\3 курс\Практика ТРПО\Прокат\БД.accdb';";

public Data\_modification\_Car()

{

InitializeComponent();

}

Point lastPoint;

private void panel1\_MouseMove(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

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

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

}

}

private void panel1\_MouseDown(object sender, MouseEventArgs e)

{

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

}

private void radioButtonAdd\_CheckedChanged(object sender, EventArgs e)

{

textBoxId.Visible = true;

textBoxStatus.Visible = true;

textBoxModel.Visible = true;

textBoxMarka.Visible = true;

textBoxColor.Visible = true;

textBoxYear.Visible = true;

textBoxTransmission.Visible = true;

labelId.Visible = true;

labelStatus.Visible = true;

labelModel.Visible = true;

labelMarka.Visible = true;

labelColor.Visible = true;

labelYear.Visible = true;

labelTransmission.Visible = true;

}

private void radioButtonUpdate\_CheckedChanged(object sender, EventArgs e)

{

textBoxId.Visible = true;

textBoxStatus.Visible = true;

textBoxModel.Visible = false;

textBoxMarka.Visible = false;

textBoxColor.Visible = false;

textBoxYear.Visible = false;

textBoxTransmission.Visible = false;

labelId.Visible = true;

labelStatus.Visible = true;

labelModel.Visible = false;

labelMarka.Visible = false;

labelColor.Visible = false;

labelYear.Visible = false;

labelTransmission.Visible = false;

}

private void radioButtonDelete\_CheckedChanged(object sender, EventArgs e)

{

textBoxId.Visible = true;

textBoxStatus.Visible = false;

textBoxModel.Visible = false;

textBoxMarka.Visible = false;

textBoxColor.Visible = false;

textBoxYear.Visible = false;

textBoxTransmission.Visible = false;

labelId.Visible = true;

labelStatus.Visible = false;

labelModel.Visible = false;

labelMarka.Visible = false;

labelColor.Visible = false;

labelYear.Visible = false;

labelTransmission.Visible = false;

}

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

{

this.Close();

}

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

{

OleDbConnection con = new OleDbConnection(str);

con.Open();

if (radioButtonAdd.Checked)

{

try

{

string request = "INSERT INTO Автомобили ([id\_автомобиля], [Статус], [Модель автомобиля], [Марка автомобиля], [Цвет], [Год выпуска], [Тип коробки передач]) VALUES (@Id, @Status, @Model, @Marka, @Color, @Year, @Transmission)";

OleDbCommand command = new OleDbCommand(request, con);

command.Parameters.AddWithValue("@Id", textBoxId.Text);

command.Parameters.AddWithValue("@Status", textBoxStatus.Text);

command.Parameters.AddWithValue("@Model", textBoxModel.Text);

command.Parameters.AddWithValue("@Marka", textBoxMarka.Text);

command.Parameters.AddWithValue("@Color", textBoxColor.Text);

command.Parameters.AddWithValue("@Year", textBoxYear.Text);

command.Parameters.AddWithValue("@Transmission", textBoxTransmission.Text);

command.ExecuteNonQuery();

MessageBox.Show("Данные успешно сохранены и обновлены в базе данных.", "Сообщение");

}

catch (Exception ex)

{

MessageBox.Show("Ошибка сохранения данных: " + ex.Message);

}

}

else if (radioButtonUpdate.Checked)

{

try

{

string request = "UPDATE Автомобили SET [Статус] = @Status WHERE [id\_автомобиля] = @Id";

OleDbCommand command = new OleDbCommand(request, con);

command.Parameters.AddWithValue("@Status", textBoxStatus.Text);

command.Parameters.AddWithValue("@Id", textBoxId.Text);

command.ExecuteNonQuery();

MessageBox.Show("Данные успешно сохранены и обновлены в базе данных.", "Сообщение");

}

catch (Exception ex)

{

MessageBox.Show("Ошибка сохранения данных: " + ex.Message);

}

}

else if (radioButtonDelete.Checked)

{

try

{

string request = "DELETE FROM Автомобили WHERE [id\_автомобиля] = @Id";

OleDbCommand command = new OleDbCommand(request, con);

command.Parameters.AddWithValue("@Id", textBoxId.Text);

command.ExecuteNonQuery();

MessageBox.Show("Данные успешно сохранены и обновлены в базе данных.", "Сообщение");

}

catch (Exception ex)

{

MessageBox.Show("Ошибка сохранения данных: " + ex.Message);

}

}

con.Close();

}

}

}

//Форма "Модификация данных состояний"

namespace Прокат

{

public partial class Data\_Modification\_Condition : Form

{

public static string str = @"Provider=Microsoft.Ace.OLEDB.12.0;Data Source='D:\Study\3 курс\Практика ТРПО\Прокат\БД.accdb';";

string comm = "Select \* from Автомобили";

public Data\_Modification\_Condition()

{

InitializeComponent();

}

Point lastPoint;

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

{

this.Close();

}

private void panel1\_MouseMove(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

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

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

}

}

private void panel1\_MouseDown(object sender, MouseEventArgs e)

{

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

}

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

{

OleDbConnection con = new OleDbConnection(str);

con.Open();

try

{

string request = "INSERT INTO Состояние ([id\_автомобиля], [Дата последнего обслуживания], [Работы]) VALUES (@Id, @Date, @Work)";

OleDbCommand command = new OleDbCommand(request, con);

command.Parameters.AddWithValue("@Id", textBoxIdCar.Text);

command.Parameters.AddWithValue("@Date", textBoxDate.Text);

command.Parameters.AddWithValue("@Work", textBoxWork.Text);

command.ExecuteNonQuery();

MessageBox.Show("Данные успешно сохранены и обновлены в базе данных.", "Сообщение");

}

catch (Exception ex)

{

MessageBox.Show("Ошибка сохранения данных: " + ex.Message);

}

}

}

}

//Форма "Модификация данных договоров"

namespace Прокат

{

public partial class Data\_modification\_Agreement : Form

{

public static string str = @"Provider=Microsoft.Ace.OLEDB.12.0;Data Source='D:\Study\3 курс\Практика ТРПО\Прокат\БД.accdb';";

//string comm = "Select \* from Аренда";

public Data\_modification\_Agreement()

{

InitializeComponent();

}

Point lastPoint;

private void panel1\_MouseMove(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

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

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

}

}

private void panel1\_MouseDown(object sender, MouseEventArgs e)

{

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

}

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

{

this.Close();

}

private void radioButtonAdd\_CheckedChanged(object sender, EventArgs e)

{

textBoxIdClient.Visible = true;

textBoxIdCar.Visible = true;

textBoxStart.Visible = true;

textBoxEnd.Visible = true;

labelIdClient.Visible = true;

labelIdCar.Visible = true;

labelStart.Visible = true;

labelEnd.Visible = true;

}

private void radioButtonDelete\_CheckedChanged(object sender, EventArgs e)

{

textBoxIdClient.Visible = false;

textBoxIdCar.Visible = true;

textBoxStart.Visible = true;

textBoxEnd.Visible = false;

labelIdClient.Visible = false;

labelIdCar.Visible = true;

labelStart.Visible = true;

labelEnd.Visible = false;

}

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

{

OleDbConnection con = new OleDbConnection(str);

con.Open();

if (radioButtonAdd.Checked)

{

try

{

string request = "INSERT INTO Аренда ([id\_клиента], [id\_автомобиля], [Дата заключения], [Дата окончания]) VALUES (@IdClient, @IdCar, @Start, @End)";

OleDbCommand command = new OleDbCommand(request, con);

command.Parameters.AddWithValue("@IdClient", textBoxIdClient.Text);

command.Parameters.AddWithValue("@IdCar", textBoxIdCar.Text);

command.Parameters.AddWithValue("@Start", textBoxStart.Text);

command.Parameters.AddWithValue("@End", textBoxEnd.Text);

command.ExecuteNonQuery();

MessageBox.Show("Данные успешно сохранены и обновлены в базе данных.", "Сообщение");

}

catch (Exception ex)

{

MessageBox.Show("Ошибка сохранения данных: " + ex.Message);

}

}

else if (radioButtonDelete.Checked)

{

try

{

string request = "DELETE FROM Аренда WHERE [id\_автомобиля] = @IdCar AND [Дата заключения] = @Start";

OleDbCommand command = new OleDbCommand(request, con);

command.Parameters.AddWithValue("@IdCar", textBoxIdCar.Text);

command.Parameters.AddWithValue("@Start", textBoxStart.Text);

command.ExecuteNonQuery();

MessageBox.Show("Данные успешно сохранены и обновлены в базе данных.", "Сообщение");

}

catch (Exception ex)

{

MessageBox.Show("Ошибка сохранения данных: " + ex.Message);

}

}

con.Close();

}

}

}

//Форма "Модификация данных цен"

namespace Прокат

{

public partial class Data\_modification\_Price : Form

{

public static string str = @"Provider=Microsoft.Ace.OLEDB.12.0;Data Source='D:\Study\3 курс\Практика ТРПО\Прокат\БД.accdb';";

//string comm = "Select \* from Автомобили";

public Data\_modification\_Price()

{

InitializeComponent();

}

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

{

this.Close();

}

Point lastPoint;

private void panel1\_MouseMove(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

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

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

}

}

private void panel1\_MouseDown(object sender, MouseEventArgs e)

{

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

}

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

{

OleDbConnection con = new OleDbConnection(str);

con.Open();

try

{

string request = "INSERT INTO Цена ([id\_автомобиля], [Дата], [Цена за час(€)]) VALUES (@Id, @Date, @Price)";

OleDbCommand command = new OleDbCommand(request, con);

command.Parameters.AddWithValue("@Id", textBoxIdCar.Text);

command.Parameters.AddWithValue("@Date", textBoxDate.Text);

command.Parameters.AddWithValue("@Price", textBoxPrice.Text);

command.ExecuteNonQuery();

MessageBox.Show("Данные успешно сохранены и обновлены в базе данных.", "Сообщение");

}

catch (Exception ex)

{

MessageBox.Show("Ошибка сохранения данных: " + ex.Message);

}

}

}

}