АННОТАЦИЯ

Отчет о курсовой работе: 79 с., 36 рис., 16 табл., 1 приложение, 8 источников.

Объект исследования – процесс предоставление справки об автомобиле.

Цель работы – разработка базы данных для приложения поиска, просмотра и редактирования информации об автомобилях.

Метод исследования – изучение принципов разработки приложений средствами .NET, принципов работы с базой данных MS SQL.

В работе были использованы технологии C#, .NET, Visual Studio 2015, Windows Form, MS SQL, Visual Studio Install Project, MS SQL Visual Studio Tools.

В результате решения задачи было разработано приложение и база данных, для получения справки об автомобилях, а также база данных для приложения.

Дальнейшее развитие программы связано с расширением ее возможностей, увеличением числа опций.

БАЗА ДАННЫХ, СУБД, MS SQL, ПРИЛОЖЕНИЕ, ADO.NET, WINDOWS FORM, VISUAL STUDIO, .NET FRAMEWORK

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее рынок продажи автомобилей достаточно развит и продолжает развиваться. С каждым днем увеличивается клиентская и дилерские базы. Дилерские компании часто подают свои автомобили в «белом свете», что мешает клиентам производить объективные оценки перед покупкой. Так же стоить отметить, что небольшие магазины, не занимающиеся распространением единственной марки часто вынуждены создавать свои ресурсы, поэтому необходимо так же развивать возможность этих самих клиентов получать объёмную, проверенную информацию по каждому автомобилю. Поэтому целью курсовой работы является разработка информативного автомобильного справочника с возможностью пользователей делиться мнением и опытом.

Исходя из указанной цели, можно выделить частные задачи, поставленные в курсовой работе:

1. Изучить нужные характеристики, которое наиболее важны при поиске и выборе автомобиля.
2. Проанализировать существующие аналоги и их реализацию.
3. В качестве результата анализа составить ТЗ для программы.
4. Изучить технологии: C#, .NET, Windows Form, MS SQL, ADO.NET, MS SQL, Visual Studio 2015, Visual Studio Project Install.
5. Разработать структуру программы и базы данных, разработать функционал программы, соответствующий техническому заданию.
6. Разработать интерфейс программы, логику программы и базу данных.

1 Анализ предметной области

1.1 Состояние вопроса

На данный момент вопрос поиска информации об автомобилях достаточно развит. В интернете есть множество сайтов, на которых можно получить данную информацию. Однако большинство данных информационных ресурсов созданы теми, перед кем стоит задача продать автомобиль, что бесспорно искажает достоверность данных.

Провелось изучение автомобильных справочников в интернете, после чего можно сказать, что большинство информационных ресурсов по данному вопросу можно разбить на две категории: сайты поставщиков и пользовательские автомобильные форумы. Для примера рассмотрим по одному примеру из каждой категории для выявления их положительных и отрицательных сторон. Из категории официальных источников был взят официальный сайт BMW в России [1], а со стороны автомобильных форумов был взят с наибольшим индексом поиска в google – aboutCar.ru [2].

Таблица 1 – BMW в России.

Положительные качества	Отрицательные качества
1. Официальная информация. 2. Предоставляет обширную информации об автомобиле при разных комплектациях. 3. Предоставляет отдельную информацию об особенностях авто (система управления жестами и т.д.)	1. Нет возможности отзыва или какой-либо оценки со стороны владельцев. 2. Содержит информацию только по данной марке. 3. Поиск только по ключевому слову и по серии марки.

Таблица 2 – aboutCar.ru

Положительные качества	Отрицательные качества
1. Возможность создания личного кабинета, в котором можно сохранять темы с интересуемыми автомобилями, ставить свое фото в профиль и так далее. 2. В наличии десятки марок и сотни популярнейших автомобилей. 3. По мимо обширному выбору есть возможность добавить авто. 4. О каждом автомобиле можно узнать множество информации из опыта владельцев.	1. Никто за достоверность информации ответственности не несет. 2. Нет поиска автомобилей (только поиск по темам и сообщениям). 3. Чтобы собрать всю информацию следует прочитать несколько тем.

По изученным образцам можно сказать, что на данный момент все виды информационных ресурсов достаточно информативны, но все-же обладают рядом недостатков, основной из которых – отсутствие быстрого и удобного поиска. Исходя из вышесказанного можно сказать, что в автомобильном справочнике должно быть:

- Удобный и тонкий поиск.
- Быстрое ознакомление с основными характеристиками автомобиля.
- Возможность просмотра мнения других пользователей и высказывания своего.
- Возможность самому добавлять автомобили в базу и следить за этими записями.

1.2 Актуальность и цель работы

В ходе анализа состояния вопроса было выявлено, что существующие аналоги обладают рядом недостатков, это значит, что создание универсального автомобильного справочника все-еще актуально. Также можно добавить, что и в общем автомобильный рынок не стоит на месте, так что и ресурсы, дающие информацию в этой среде должны развиваться.

Внедрение разрабатываемого автомобильного справочника позволит пользователю:

1. Производить поиск автомобилей по разным характеристикам.
2. Составлять мнение об автомобиле на основе его характеристик, описания и оценках других пользователей.
3. Добавлять и редактировать свои записи об автомобилях, а также следить за добавленными и оцененными записями.

Целью данной курсовой работы является разработка приложения «Автомобильный справочник». Исходя из указанной цели можно составить частные задачи, поставленные в курсовой работе:

1. Изучить нужные характеристики, которое наиболее важны при поиске и выборе автомобиля.
2. Изучить технологии: C#, .NET, Windows Form, MS SQL, Visual Studio 2015, ADO.NET.
3. Разработать структуру программы и базы данных, разработать функционал программы, соответствующий техническому заданию.
4. Разработать интерфейс программы, логику программы и базу данных.

2 Техническое задание

2.1 Общие требования к продукту

Поскольку приложение служит для информирования пользователя об автомобилях, то следует учесть все основные черты автомобилей, которые интересуют пользователе при поиске. Такими характеристиками были выделены:

- Марка
- Тип кузова
- Коробка передач
- Привод
- Максимальная скорость
- Средняя цена

По мимо выделенных характеристик запись должна содержать имя автомобиля, его описание, а также среднюю оценку пользователей и подробный список с оценками каждого пользователя.

Поскольку, выложенную информацию пользователей будут видеть другие пользователи, то стоит ограничить права пользователей, а именно поделить типы пользователей на:

- Пользователь
- Доверенный пользователь/Модератор
- Администратор

Права обычного пользователя должны ограничиваться на просмотре информации об автомобилях и возможности оценивать другие автомобили. Права модератора включают возможность добавления записей об автомобилях и их редактировании, а права администратора должна быть возможность удалять чужие комментарии, записи, аккаунты и менять права пользователей.

У каждого пользователя дожжён быть свой кабинет, в котором можно будет менять информацию о себе. Также в кабинете должна быть возможность следить за всеми выложенными записями и комментариями. Должна быть возможность посещать не только свой, но и чужой кабинет, однако просматривать чужие комментарии имеет право только администратор приложения.

Информация о пользователе должна содержать:

- Имя
- Фотографию
- Роль на сайте
- Дата регистрации

При запуске приложения пользователя встречает окно входа в аккаунт, из него пользователь может зайти в приложения или вызвать окно регистрации. После успешного ввода логина и пароля открывается главное окно, в котором, по мимо всего, содержится информация о текущем пользователе.

Не смотря на богатый функционал интерфейс должен быть понятным и компактным, с наличием окна справки и подсказками по мере использования приложения.

2.2 Позиционирование продукта

2.2.1 Требование к пользовательским интерфейсам

Поскольку приложение обладает богатой функциональностью и при этом должно быть компактным, то можно выдвинуть следующие требования к дизайну приложения:

1. При надобности заменять надписи на кнопках на графические иконки.
2. Прятать от пользователя недоступные для него элементы.

3. При наведении на элементы интерфейса должны появляться подсказки.

Интерфейс приложения, в соответствии с функциональностью, должен быть визуально разбит на следующие логические блоки:

- Блок поиска записей
- Информация о текущем пользователе
- Блок с управляющими элементами
- Блок с информацией о пользователе (кабинет пользователя)
- Блок просмотра и редактирования информации о машине
- Блок просмотра результатов поиска
- Блок с просмотром лучших и новейших записей

Логично было бы организовать последние четыре элемента списка так, чтобы просматривать только один из них. Данное действие позволить сохранить место на главном окне.

На основе вышесказанного можно составить требования к интерфейсу приложения и его дочерним окнам, представляемым пользователям:

Главное окно (рисунок 1).

1. Главное окно
2. Главная панель приложения
3. Блок, содержащий дочерние окна (окно личного кабинета, окно просмотра информации об авто и т.д.)
4. Панель поиска
5. Фотография пользователя
6. Ник пользователя
7. Элементы управления приложением
8. Кнопка обновления
9. Кнопка перехода к странице новых и лучших авто
10. Кнопка добавления новой записи
11. Кнопка перехода к профилю текущего пользователя
12. Кнопка вызова справки
13. Кнопка выхода из аккаунта
14. Кнопка выхода из приложения

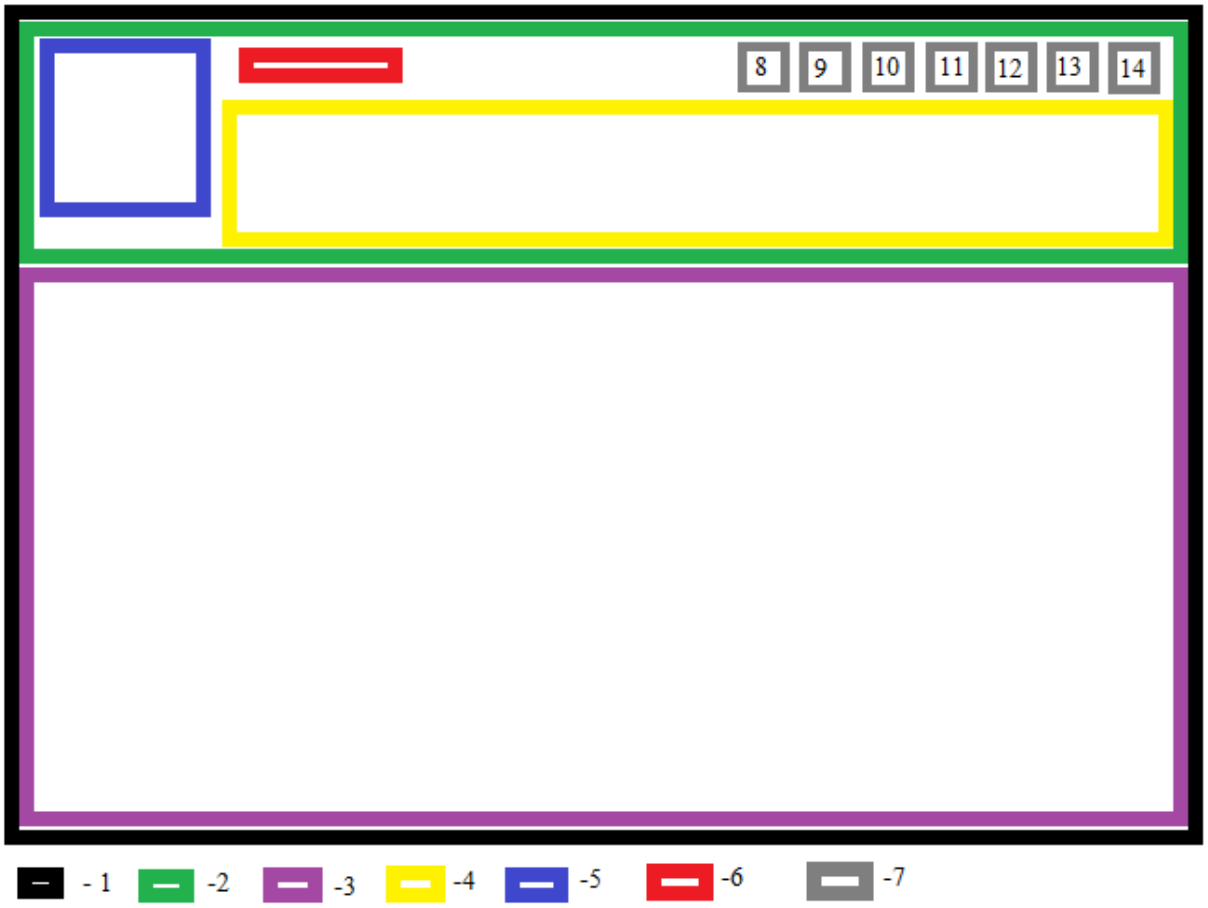


Рисунок 1

При отсутствии фотографии у аккаунта должна отображаться стандартная иконка аккаунта.

Окно просмотра лучших и новейших записей (рисунок 2).

1. Новейшие записи об автомобилях
2. Лучшие записи об автомобилях
3. Кнопка «Еще»

В списках должно отображаться ограниченное количество элементов, но при желании пользователя сможет добавить элементов в список с помощью кнопки «Еще».

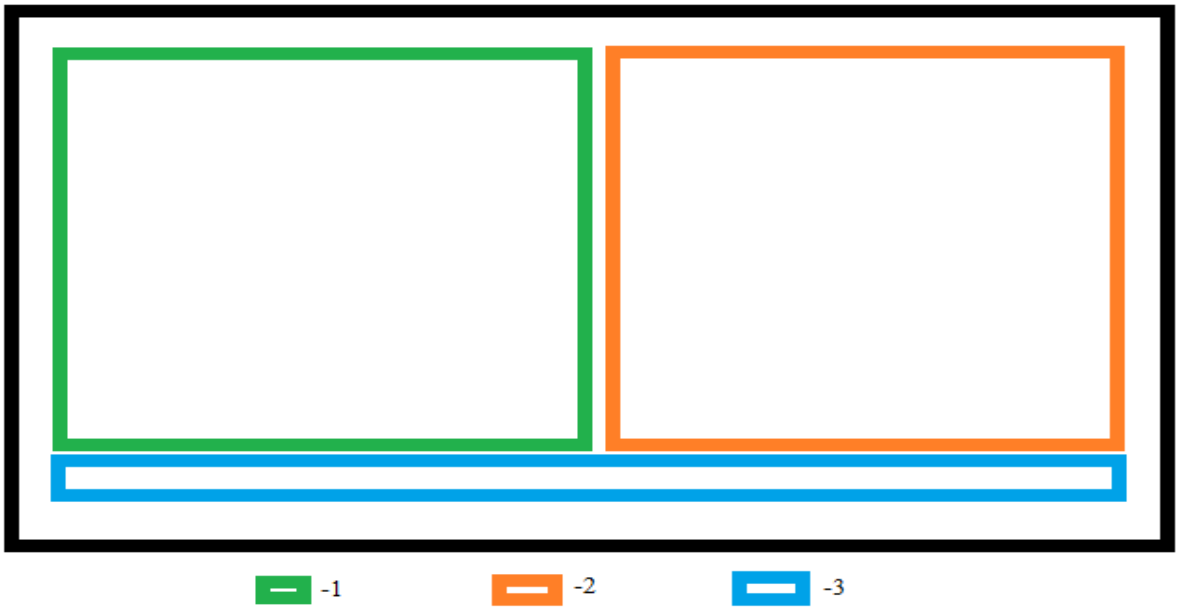


Рисунок 2

Окно просмотра и редактирования информации об автомобиле (рисунок 3).

1. Панель редактирования записи
2. Панель характеристик автомобиля
3. Панель комментариев
4. Описание автомобиля
5. Фотография автомобиля
6. Панель комментария текущего пользователя
7. Текстовое поле для ввода комментария
8. Выбор оценки для комментария
9. Кнопка отправки комментария
10. Панель, в которой будет список комментариев
11. Кнопка добавления комментариев к списку
12. Кнопка редактирования записи
13. Кнопка сохранения записи
14. Кнопка удаления записи
15. Название автомобиля
16. Рейтинг автомобиля в приложении
17. Панель с характеристиками автомобилей

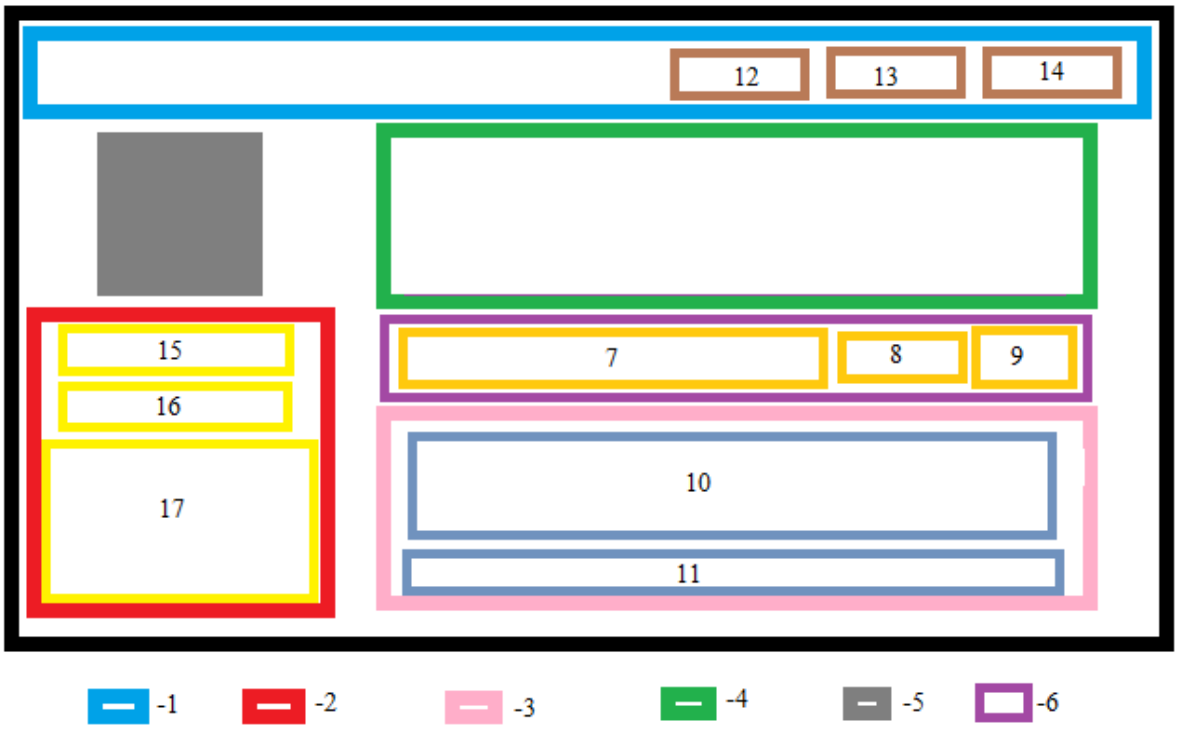


Рисунок 3

Каждая кнопка на панели редактирования скрыта, если у пользователя нет прав на ее использование. Если недоступна ни одна из команд редактирования записи сама панель тоже скрыта. Если пользователь уже комментировал запись – вместо панели комментария пользователя отображается его комментарий. Рейтинг автомобилей должен отображаться графически (10 балов- 10 звезд).

Окно кабинета пользователя (рисунок 4).

1. Панель редактирования
2. Панель с информацией о пользователе
3. Все записи пользователя
4. Все комментарии пользователя
5. Фотография пользователя
6. Список комментариев пользователя
7. Кнопка добавления комментариев к списку комментариев пользователя
8. Список записей пользователя

9. Кнопка добавления записей к списку записей пользователя
10. Имя пользователя
11. Обозначение прав пользователя
12. Дата регистрации пользователя
13. Кнопка удаления аккаунта
14. Кнопка сохранения записи
15. Кнопка редактирования записи

Панель редактирования работает аналогично панели редактирования записи автомобиля.

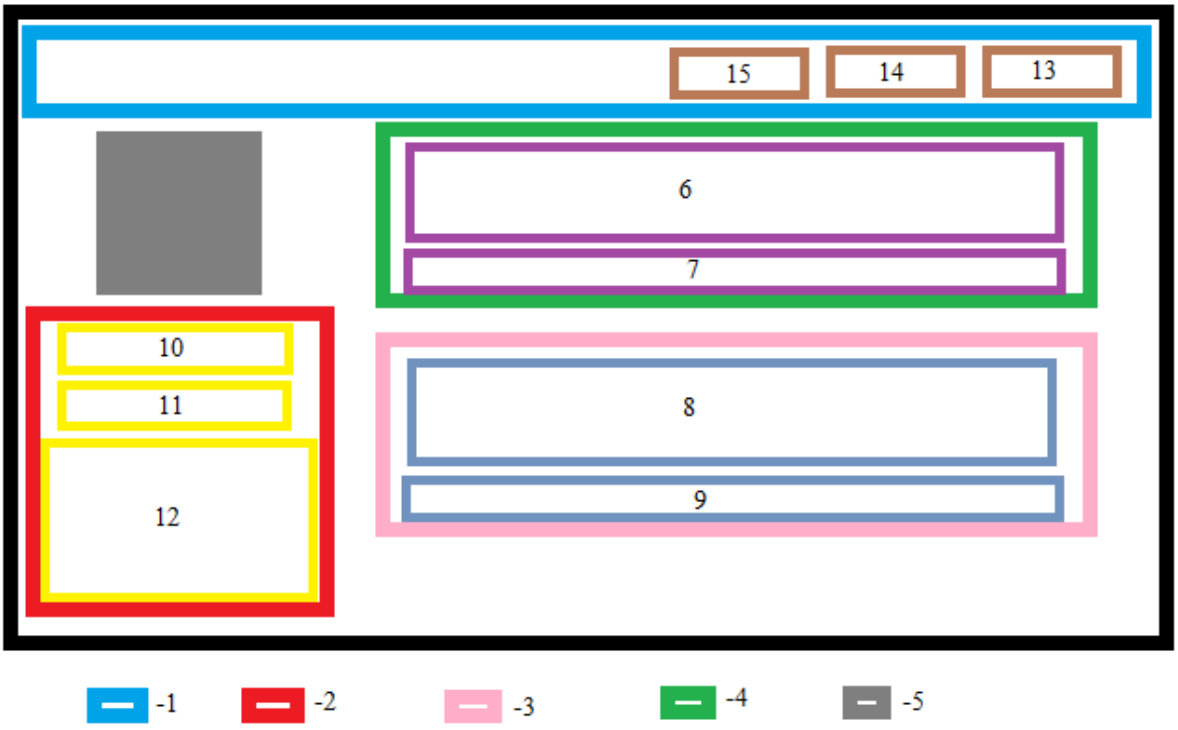


Рисунок 4

Окно просмотра результатов поиска (рисунок 5).

1. Список записей, удовлетворяющих поиску
2. Управление найденными записями

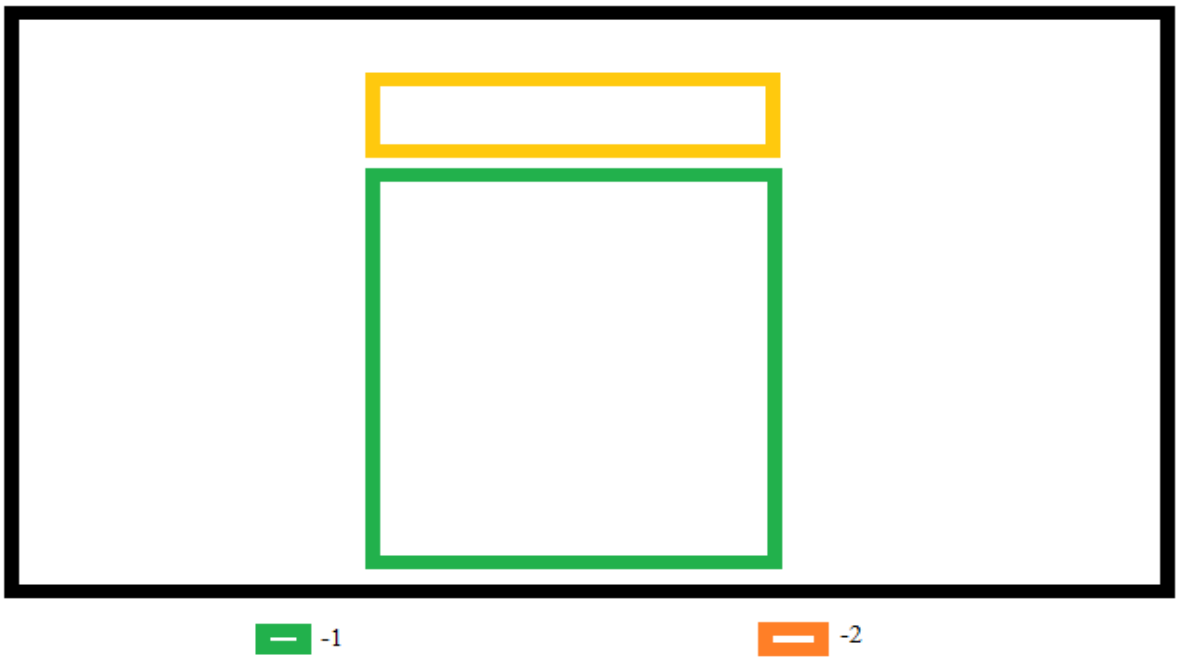


Рисунок 5

Окно входа в аккаунт (рисунок 6).

Рисунок 6

1. Элементы для ввода логина и пароля
2. Кнопка входа
3. Вызов окна регистрации
4. Справка
5. Выход

Окно регистрации (рисунок 7).

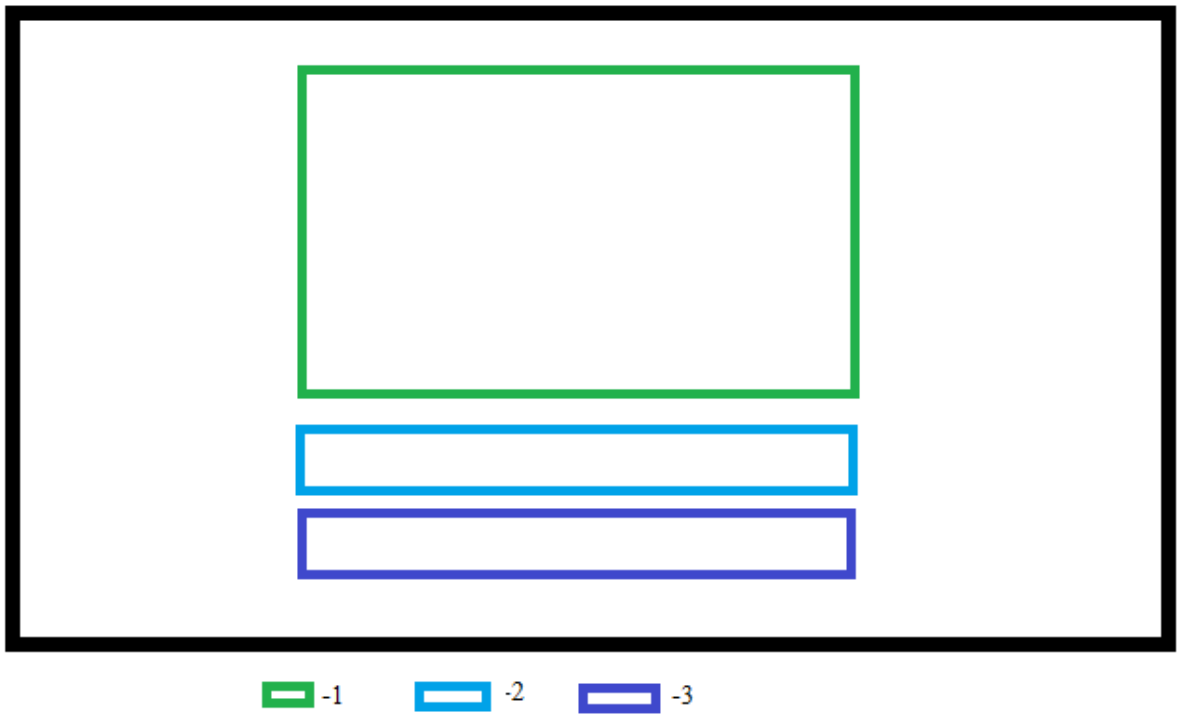


Рисунок 7

1. Элементы для ввода логина
2. Кнопка регистрации
3. Кнопка отмены

Панель поиска (рисунок 8).

1. Выбор марки
2. Выбор кузова

3. Выбор привод
4. Выбор коробки передач
5. Выставление диапазона цен
6. Выставление диапазона скорости
7. Выбор ключевого слова
8. Кнопка начала поиска

Рисунок 8

Вид комментария (рисунок 9).

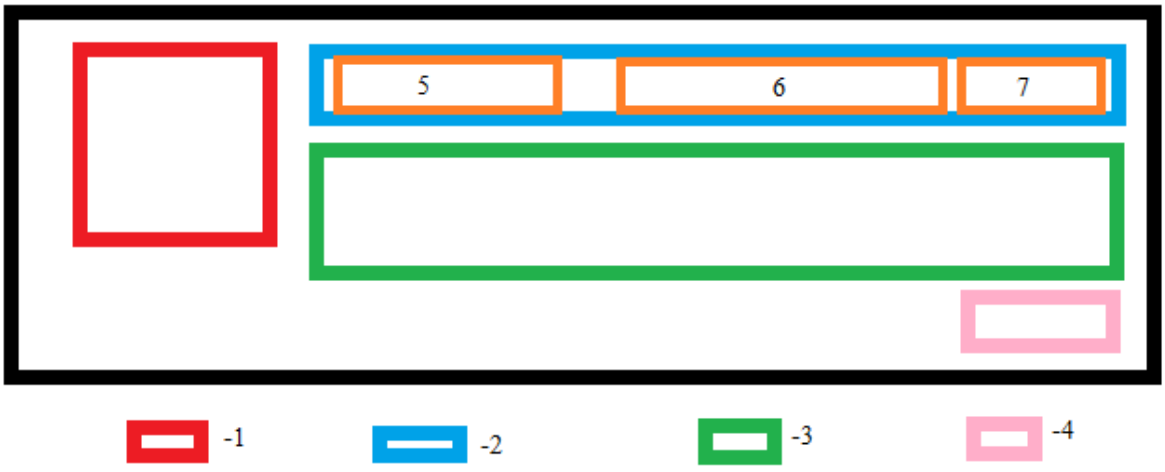


Рисунок 9

1. Фотография пользователя, сделавшего комментарий
2. Панель с информацией о комментарии

3. Панель с содержанием комментария
4. Кнопка удаления комментария
5. Имя комментатора
6. Графическая оценка комментария
7. Дата комментария

Кнопка удаления видна только для владельцев комментария и владельцев.

Вид информации об авто в списках (рисунок 10).

1. Фотография автомобиля
2. Название автомобиля
3. Панель с содержанием некоторых характеристик
4. Кнопка «перейти к записи»
5. Отображение скорости
6. Отображение коробки передач
7. Отображение привода
8. Отображение средняя оценка

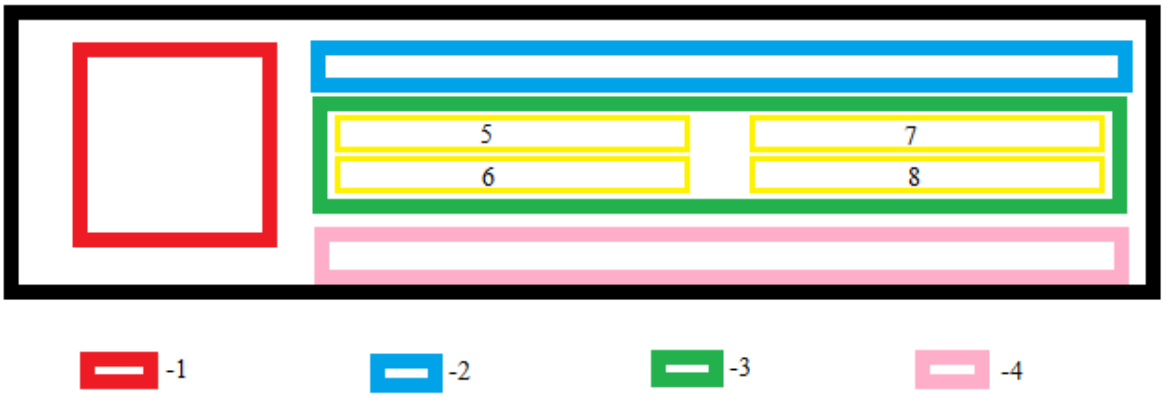


Рисунок 10

Окно справки (рисунок 11).

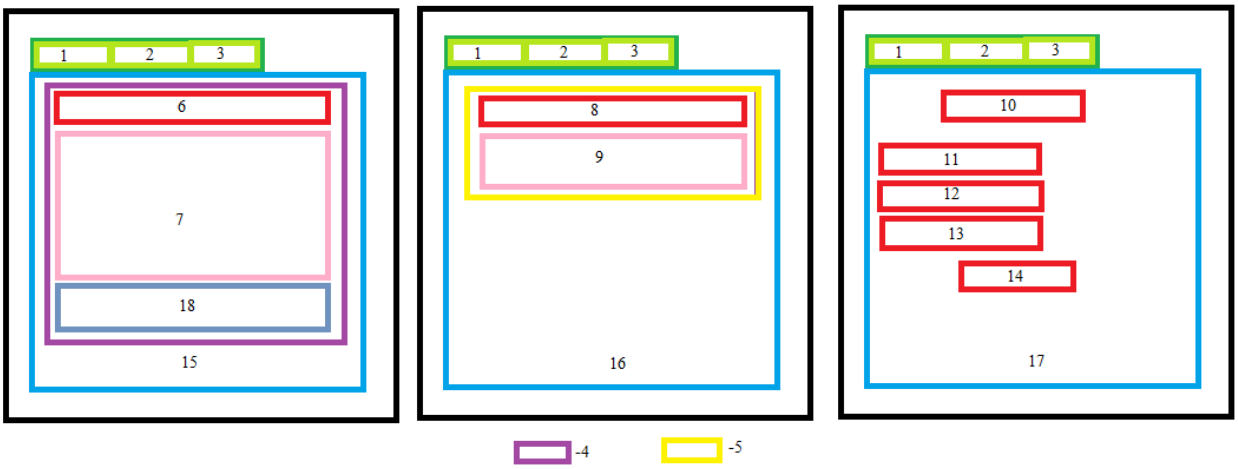


Рисунок 11

1. Кнопка открытия вкладки «Введение»
2. Кнопка открытия вкладки «F.A.Q.»
3. Кнопка открытия вкладки «Об приложении»
4. Запись вкладки «Введение»
5. Запись вкладки «F.A.Q»
6. Название записи вкладки «Введение».
7. Картинка, показывающая часть интерфейса приложения
8. Часто задаваемый вопрос
9. Ответ
10. Название приложения
11. Имя автора
12. Электронный ящик автора для связи
13. Краткое описание программы
14. Место и год создания

Вкладка «Введение» содержит записи, объясняющие пользователю, как использовать программу. В записи есть название записи, изображение и описание функциональности зоны интерфейса. Вкладка «F.A.Q.» содержит список часто задаваемых вопросов и содержащимся ответом под ними.

2.2.2 Требования к программным интерфейсам

Код программы должен быть понятным, содержать комментарии, не содержать повторяющегося кода для дальнейшей разработки. Создать классы для работы с базой данных и класс, содержащий логику.

2.2.3 Требования к базе данных

База данных должна содержать данные о пользователях для успешного входа, полную информацию о машинах и пользователях. Также потребуется обеспечить связи между таблицами. База данных должна хранить большие объемы данных и обеспечить удовлетворительную скорость работы. Для создания базы данных в данном курсовом проекте была выбрана СУБД MS SQL.

2.2.4 Требования к пользователям продукта

- Подключение к сети Интернет
- Базовый навык работы с компьютером

2.3 Функции продукта

Основной функционал продукта:

1. Возможность гибкого поиска.
2. Просмотр детальной и минимальной информации о автомобиле.
3. Получение справки.
4. Добавление, удаление и редактирование записей об автомобиле.
5. Создание аккаунта и вход на существующие. Управление и удаление аккаунта, редактирование некоторых данных пользователя.
6. Разграничение пользователей по правам доступа.

7. Создание и удаление комментариев с оценкой.

Графически, функциональная схема выглядит следующим образом (рисунок 12):

Рисунок 12

2.4 Сценарии использования продукта

Начальный сценарий: Пользователь запускает приложение, открывается окно входа в аккаунт.

Сценарии окна входа в аккаунт

Сценарий 1: Пользователь вводит верные логин/пароль и нажимает кнопку входа, происходит успешная авторизация. Открывается главное окно приложения.

Сценарий 2: Пользователь неверно вводит логин/пароль или не вводит данные вовсе и нажимает кнопку входа. Появляется окно, предупреждающее

о неверном логине или пароле.

Сценарий 3: Пользователь верно вводит логин/пароль и нажимает на кнопку входа, авторизация не происходит. Появляется окно, которое оповещает пользователя об отсутствии подключения с базой.

Сценарий 4: Пользователь нажимает кнопку регистрации, открывается окно регистрации.

Сценарий 5: Пользователь нажимает кнопку выхода. Приложение завершает свою работу.

Сценарий 6: Пользователь нажимает кнопку справки. Открывается окно справки.

Окно регистрации

Сценарий 1: пользователь вводит данные и нажимает кнопку регистрации.

Сценарий 1.1: данные проходят проверку, регистрация проходит успешно. Пользователь возвращается к странице входа.

Сценарий 1.2: данные проходят проверку, но пользователь не зарегистрирован. Появляется диалог, оповещающий об ошибке регистрации со стороны приложения.

Сценарий 1.3: данные не проходят проверку, о чем пользователя оповещает появившийся диалог.

Сценарий 2: пользователь нажимает кнопку справки, открывается модальное окно справки.

Сценарий 3: пользователь нажимает кнопку отмены. Окно регистрации закрывается, пользователь возвращается к окну входа в аккаунт.

Окно справки

Сценарий 1: пользователь нажимает на кнопку «Введение», открывается вкладка «Введение».

Сценарий 2: пользователь нажимает на кнопку «F.A.Q.», открывается вкладка «F.A.Q.».

Сценарий 3: пользователь нажимает на кнопку «Об приложении», открывается вкладка «Об приложении».
Сценарий 4: пользователь нажимает на закрытие окна, окно закрывается.

Главное окно приложения
Сценарий 1: пользователь нажимает на кнопку закрытия окна, приложение закрывается.
Сценарий 2: пользователь нажимает на кнопку выходи аз аккаунта, текущее окно закрывается и открывается окно входа в аккаунт.
Сценарий 3: пользователь нажимает на кнопку вызова справки, открывается модальное окно справки.
Сценарий 4: пользователь нажимает кнопку открытия вкладки просмотра личного кабинета и открывается вкладка просмотра аккаунта текущего пользователя.
Сценарий 5: пользователь нажимает на кнопку создания записи об авто.
Сценарий 5.1: пользователь имеет не имеет прав на создание – появляется окно, предупреждающее об этом.
Сценарий 5.2: открывается вкладка редактирования записи об автомобиле.
Сценарий 6: пользователь нажимает на кнопку открытия вкладки просмотра лучших и новейших записей, после чего данная вкладка открывается.
Сценарий 7: пользователь нажимает на вкладку обновления. Обновляются все элементы текущего главного окна, включая текущую вкладку.
Сценарий 8: пользователь вводит данные для поиска и нажимает на кнопку поиска, открывается вкладка с результатами поиска.

Вкладка новейших и лучших записей
Сценарий 1: пользователь нажимает кнопку «Еще». Список обновляется и становится на пять элементов больше
Сценарий 2: пользователь нажимает кнопку «подробнее» на любом из элементов списка и открывается вкладка просмотра и редактирования данной

записи.

Вкладка просмотра и редактирования записи об авто

Сценарий 1: пользователь нажимает кнопку «Редактировать». Элементы с содержанием имени изменяется на редактируемый, а у фотографии появляется кнопка загрузки фото. У администраторов появляется возможность поменять права пользователя.

Сценарий 1.1: пользователь нажимает на кнопку изменения фотографии. Появляется диалог выбора файла. Пользователь выбирает файл и жмет кнопку «ок». Диалог закрывается, а в панели фотографии изображение меняется на выбранное.

Сценарий 2: пользователь нажимает кнопку «Сохранить», данные с элементов изменяют параметры пользовательского аккаунта.

Сценарий 3: пользователь нажимает кнопку «удалить». Запись удаляется, а пользователя перемещают во вкладку просмотра новых и лучших записей.

Сценарий 4: пользователь не комментировал запись. Пользователь нажимает на кнопку отправки комментария, страница обновляется.

Сценарий 5: пользователь комментировал запись. Вместо панели отправки комментария находится комментарий пользователя.

Сценарий 5.1: пользователь нажимает на кнопку своего комментария «Удалить». Комментарий удаляется, а вкладка перезагружается.

Сценарий 6: пользователь нажимает на имя создателя записи, открывается вкладка просмотра пользователя, создавшего запись.

Вкладка просмотра личного кабинета пользователя

Сценарий 1: пользователь нажимает кнопку «Редактировать». Элементы с содержанием имени изменяется на редактируемый, а у фотографии появляется кнопка загрузки фото. У администраторов появляется возможность поменять права другого пользователя.

Сценарий 2: пользователь нажимает кнопку «Сохранить», данные с

элементов изменяют параметры пользовательского аккаунта.

Сценарий 3: пользователь нажимает кнопку «удалить». Аккаунт удаляется, а пользователя перемещают в окно входа в аккаунт.

Сценарий 4: пользователь жмет удалить на любой из своих комментариев. Комментарий удаляется, а окно обновляется.

Сценарий 5: пользователь нажимает кнопку «подробнее» на любом из элементов списка и открывается вкладка просмотра и редактирования данной записи.

Сценарий 6: пользователь нажимает на кнопку еще в панели комментариев или в панели записей, и данная запись увеличивается на пять элементов.

Вкладка просмотра результатов поиска

Сценарий 1: пользователь изменяет данные в элементах поиска и повторно жмет кнопку поиска. К списку результатов поиска добавляются результаты нового поиска.

Сценарий 2: пользователь нажимает на кнопку «очистить», и список с результатами поиска очищается.

2.5 Функциональные требовании

Таблица 3 – Сценарий регистрации аккаунта

Сценарий использования	Регистрация
Приоритет	Важно/Желательно/Необязательно
Триггер	Нажатие на кнопку «Регистрация»
Предусловие	Открытие формы регистрации Корректное заполнение обязательных полей формы

Окончание таблицы 3

Основной сценарий	Пользователь уведомлен об успешной регистрации
Постусловие	Добавление нового пользователя в базу данных Перенаправление пользователя в окно входа
Сценарий исключительных ситуаций	Пользователь не добавлен в базу, появляется окно с предупреждением.

Таблица 4 – Сценарий авторизации пользователя

Сценарий использования	Вход в систему
Приоритет	Важно/Желательно/Необязательно
Триггер	Нажатие на кнопку «Вход»
Предусловие	Открытие формы входа Корректное заполнение обязательных полей формы
Основной сценарий	Проверка пользователя в базе
Постусловие	Открывается главное окно приложения
Сценарий исключительных ситуаций	Пользователя нет в базе, появляется окно с предупреждением.

Таблица 5 – Сценарий просмотра информации об автомобиле

Сценарий использования	Просмотр информации об автомобиле
Приоритет	Важно/Желательно/Необязательно
Триггер	Нажатие на кнопку «Подробнее» в пользовательской форме приложения
Предусловие	-

Окончание таблицы 5

Основной сценарий	Считывание данных с базы Заполнение элементов вкладки
Постусловие	Открывается заполненная вкладка просмотра информации об автомобиле
Сценарий исключительных ситуаций	Запись была удалена из базы или база не отвечает. Появляется окно с предупреждением.

Таблица 6 – Сценарий редактирования информации об автомобиле

Сценарий использования	Изменение информации об автомобиле
Приоритет	Важно/Желательно/Необязательно
Триггер	Нажатие на кнопку «Редактировать»
Предусловие	Открытие вкладки просмотра записи об авто
Основной сценарий	Считывание данных с базы Изменение и повторное заполнение элементов вкладки Редактирование пользователем
Постусловие	Считывание данных с элементов управления Сохранение данных в базе данных Возвращение вкладки к виду просмотра
Сценарий исключительных ситуаций	Запись была удалена из базы или база не отвечает. Появляется окно с предупреждением.

Таблица 7 – Сценарий удаления аккаунта

Сценарий использования	Удаление аккаунта
Приоритет	Важно/Желательно/Необязательно
Триггер	Нажатие на кнопку «Удалить» на панели редактирования личного кабинета (или кабинета пользователя, если вы-администратор)
Предусловие	Открытие вкладки просмотра записи об авто
Основной сценарий	Считывание данных с базы Изменение и повторное заполнение элементов вкладки Редактирование пользователем
Постусловие	Считывание данных с элементов управления Сохранение данных в базе данных Возвращение вкладки к виду просмотра
Сценарий исключительных ситуаций	Запись была удалена из базы или база не отвечает. Появляется окно с предупреждением.

3. Реализация программного продукта

3.1 Обоснование средств разработки

3.1.1 Visual Studio

Microsoft Visual Studio — линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework и Silverlight.

Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных. Visual Studio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода (как, например, Subversion и Visual SourceSafe), добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода на предметно-ориентированных языках программирования) или инструментов для прочих аспектов процесса разработки программного обеспечения (например, клиент Team Explorer для работы с Team Foundation Server).

Одним из используемых инструментов Visual Studio в данной КР является инструмент «Microsoft Visual Studio 2015 Installer Projects». Данный инструмент позволит создавать установочный файл для данного приложения.

3.1.2 .NET Framework

.NET Framework — программная платформа, выпущенная компанией Microsoft в 2002 году. Основой платформы является общеязыковая среда исполнения Common Language Runtime (CLR), которая подходит для разных языков программирования. Функциональные возможности CLR доступны в любых языках программирования, использующих эту среду.

Программа для .NET Framework, написанная на любом поддерживаемом языке программирования, сначала переводится компилятором в единый для .NET промежуточный байт-код Common Intermediate Language (CIL). В терминах .NET получается сборка, англ. assembly. Затем код либо исполняется виртуальной машиной Common Language Runtime (CLR), либо транслируется утилитой NGen.exe в исполняемый код для конкретного целевого процессора. Данные процессы видны на рисунке 13.

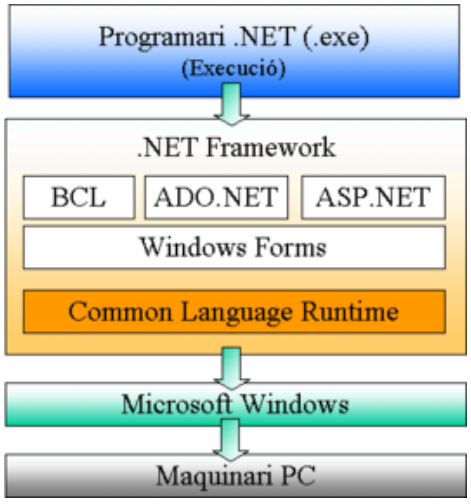


Рисунок 13

Использование виртуальной машины предпочтительно, так как избавляет разработчиков от необходимости заботиться об особенностях аппаратной части. В случае использования виртуальной машины CLR встроенный в неё JIT-компилятор «на лету» (just in time) преобразует промежуточный байт-код в машинные коды нужного процессора. Современная технология динамической компиляции позволяет достигнуть высокого уровня быстродействия. Виртуальная машина CLR также сама заботится о базовой безопасности, управлении памятью и системе исключений, избавляя разработчика от части работы.

3.1.3 Windows Forms

Windows Forms — интерфейс программирования приложений (API), отвечающий за графический интерфейс пользователя и являющийся частью Microsoft .NET Framework. Данный интерфейс упрощает доступ к элементам интерфейса Microsoft Windows за счет создания обёртки для существующего Win32 API в управляемом коде. Причём управляемый код — классы, реализующие API для Windows Forms, не зависят от языка разработки. То есть программист одинаково может использовать Windows Forms как при написании ПО на C#, C++, так и на VB.Net, J# и др.

3.1.4 ADO.NET

Платформа .NET определяет ряд пространств имен, которые позволяют непосредственно взаимодействовать с локальными и удаленными базами данных. Вместе эти пространства имен известны как ADO.NET. Библиотеки ADO.NET можно применять тремя концептуально различными способами: в подключенном режиме, в автономном режиме и с помощью технологии Entity Framework. При использовании подключенного уровня (connected layer), кодовая база явно подключается к соответствующему хранилищу

данных и отключается от него. При таком способе использования ADO.NET обычно происходит взаимодействие с хранилищем данных с помощью объектов подключения, объектов команд и объектов чтения данных.

Таблица 8 – Основные объекты ADO.NET

Тип объекта	Базовый класс	Соответствующие интерфейсы	Назначение
Connection	DbConnection	IDbConnection	Позволяет подключаться к хранилищу данных и отключаться от него. Кроме того, объекты подключения обеспечивают доступ к соответствующим объектам транзакций
Command	DbCommand	IDbCommand	Представляет SQL-запрос или хранимую процедуру. Кроме того, объекты команд предоставляют доступ к объекту чтения данных конкретного поставщика данных
DataReader	DbDataReader	IDataReader, IDataRecord	Предоставляет доступ к данным только для чтения в прямом направлении с помощью курсора на стороне сервера

Окончание таблицы 9

DataAdapter	DbDataAdapter	IDataAdapter, IDbDataAdapter	Пересылает наборы данных из хранилища данных к вызывающему процессу и обратно. Адаптеры данных содержат подключение и набор из четырех внутренних объектов команд для выборки, вставки, изменения и удаления информации в хранилище данных
Parameter	DbParameter	IDataParameter, IDbDataParameter	Представляет именованный параметр в параметризованном запросе
Transaction	DbTransaction	IDbTransaction	Инкапсулирует транзакцию в базе данных

В ADO.NET термин "объект подключения" на самом деле относится к конкретному типу, порожденному от DbConnection; объекта подключения "вообще" нет. То же можно сказать и об "объекте команды", "объекте адаптера данных" и т.д. По соглашению имена объектов в конкретном поставщике данных имеют префиксы соответствующей СУБД (например, SqlConnection, OracleConnection, SqlDataReader и т.д.).

3.1.5 Использование языка программирования C#

В данной курсовой работе был выбран благодаря тому, что язык имеет компонент Windows Form, может быть использован в .Net Framework, и в общем позволяет быстро создавать приложения под windows и не только.

По мимо вышесказанных достоинств, ЯП C# имеет ряд других преимуществ. Вот, например, преимущества C# по книге Биллига:

- создавался параллельно с каркасом Framework .NET и в полной мере учитывает все его возможности - как FCL, так и CLR;
- является полностью объектно-ориентированным языком, где даже типы, встроенные в язык, представлены классами;
- является мощным объектным языком с возможностями наследования и универсализации;
- является наследником языков C/C++, сохраняя лучшие черты этих популярных языков программирования. Общий с этими языками синтаксис, знакомые операторы языка облегчают переход программистов от C++ к C#;
- сохранив основные черты своего великого родителя, язык стал проще и надежнее. Простота и надежность, главным образом, связаны с тем, что на C# хотя и допускаются, но не поощряются такие опасные свойства C++ как указатели, адресация, разыменование, адресная арифметика;
- благодаря каркасу Framework .NET, ставшему надстройкой над операционной системой, программисты C# получают те же преимущества работы с виртуальной машиной, что и программисты Java. Эффективность кода даже повышается, поскольку исполнительная среда CLR представляет собой компилятор промежуточного языка, в то время как виртуальная Java-машина является интерпретатором байт-кода;
- мощная библиотека каркаса поддерживает удобство построения различных типов приложений на C#, позволяя легко строить Web-

службы, другие виды компонентов, достаточно просто сохранять и получать информацию из базы данных и других хранилищ данных;

- реализация, сочетающая построение надежного и эффективного кода, является немаловажным фактором, способствующим успеху C#.

3.1.6 SQL Server Data Tools

SQL Server Data Tools (SSDT) - это окончательное имя продукта, который ранее назывался SQL Server Developer Tools с рабочим названием «Juneau». Средства SSDT предоставляют передовые возможности работы для разработчиков баз данных SQL Server и SQL Azure. Появление SQL Server Data Tools (SSDT) изменило разработку баз данных благодаря внедрению универсальной декларативной модели, охватывающей все этапы разработки базы данных в среде Visual Studio (рисунок 14). Возможности SSDT Transact-SQL помогают в сборке, отладке, обслуживании и реструктурировании баз данных. Можно работать как с проектом базы данных, так и непосредственно с подключенным экземпляром базы данных (как на собственной площадке, так и в облаке).

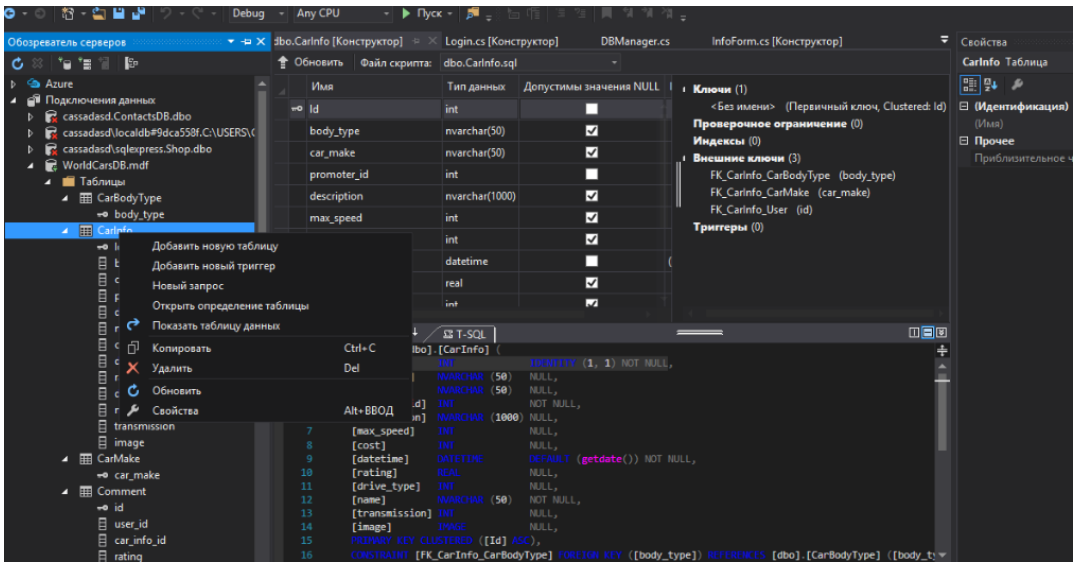


Рисунок 14

3.2 Проектирование БД

3.2.1 Концептуальное проектирование

Назначение концептуального представления БД состоит в том, чтобы представить формализованную информацию о предметной области таким образом, чтобы она было достаточно емкой для оценки глубины и корректности проработки проекта базы данных.

Были выделены основные сущности, необходимые для решения задачи:

1. Пользователь
2. Запись с информацией об автомобиле
3. Комментарий

Основными сущностями БД, как можно заметить, является запись об автомобиле и пользователь. Объект, представляющий пользователя должен содержать атрибуты для содержания: логина и пароля, даты регистрации, уровня доступа, имени внутри приложения.

Комментарий должен быть одновременно привязан к пользователю и записи об авто и содержать оценку и текст комментария к записи, а также дату и время написания.

Объект записи об авто должен содержать следующие поля: тип кузова, марка, стоимость, максимальная скорость, рейтинг, тип коробки передач, название, дата создания записи, привод и изображение машины. Запись должна быть привязана к тому, кто ее создал, чтобы знать, кто сможет редактировать ее в будущем.

На рисунке 15 представлена концептуальная схема базы данных, которая описывает сущности-объекты, необходимые для решения поставленной задачи в данной курсовой работе.

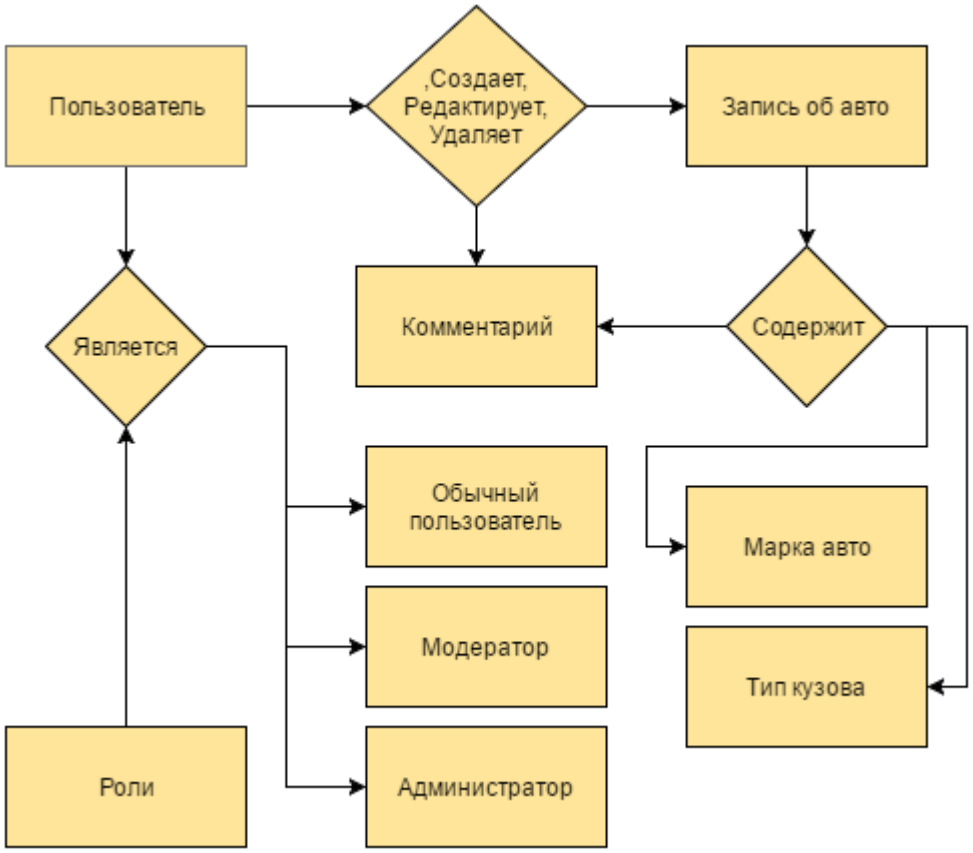


Рисунок 15

3.2.2 Логическое проектирование

При концептуальном проектировании было выделено три объекта, соответственно, в базе должны быть три таблицы, однако стоит заметить, что для объекта автомобиля есть поля, которые часто будут повторятся (тип кузова и марка авто), поэтому есть смысл отделить эти атрибуты, как отдельные таблицы, к которым будет привязана таблица записи об авто. Таким образом, к основным сущностям БД можно добавить еще две таблицы:

- CarBodyType (Тип кузова)
- CarMake (марка машины).

На рисунке 16 представлена логическая схема базы данных для курсового проекта. Для построения данной схемы было выбрано веб-приложение DbDesigner.

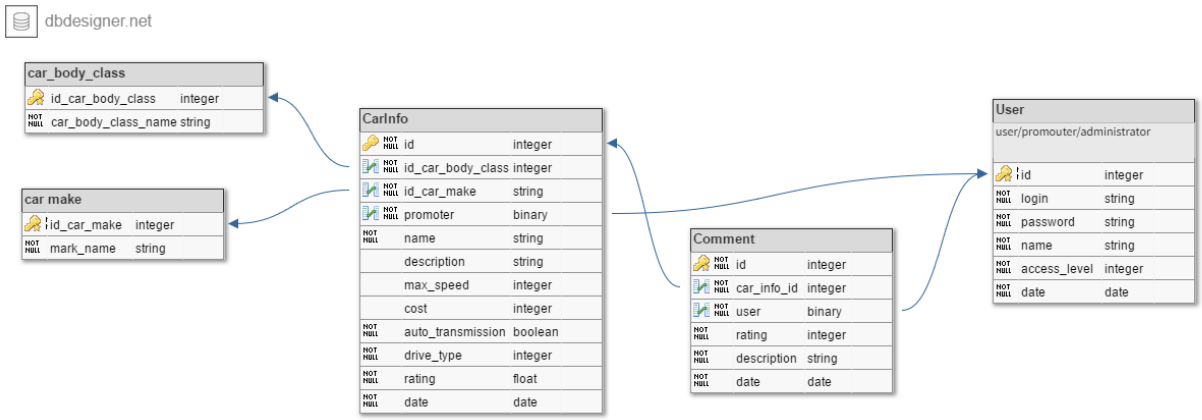


Рисунок 16

3.2.3 Физическое проектирование

Создание базы данных из логической модели, в данном случае, не имеет каких-то ярковыраженных особенностей. Создаются пять таблиц и заполняются атрибутами соответствующих типов, как и было описано в логической модели. Разница лишь в названиях типов (integer в схеме и int в СУБД или string в схеме и nvarchar в СУБД).

Первичные ключи имеют тип int, для них установленно свойство индетификатора с начальным значем в 1 и аналогичным шагом, это значит, что при добавлении новой записи, ее первичный ключ будет на единицу больше, чем предидущий, а самая первая запись в таблице будет с индетификатором равным 1.

Также, важной особенностью данной БД является то, что рейтинг в записи об авто зависит от всех комментариев. Можно было бы убрать этот атрибут и вычислять его каждый раз при создании выборки автообилей, однако при большем количестве комментариев это будет слишком ресурсоемкая операция. Лучшим решением будет не убирать данный атрибут, а поставить ему стандартное значение равным 0 и пересчитывать его каждыйраз, когда создаются, изменяются или удаляются комментарии к данной записи об автомобиле, т.е. создать триггер.

3.3 Моделирование бизнес-процессов

Моделирование бизнес-процессов является крайне важным этапом разработки, так как позволяет оценить работу приложения и в дальнейшем оптимизировать его работу.

Для создания и описание бизнес-процессов была создана блок-схема (рисунок 17), которая в полной мере описывает процессы, которые должны происходить в приложении и какие из этого должны получится результаты.

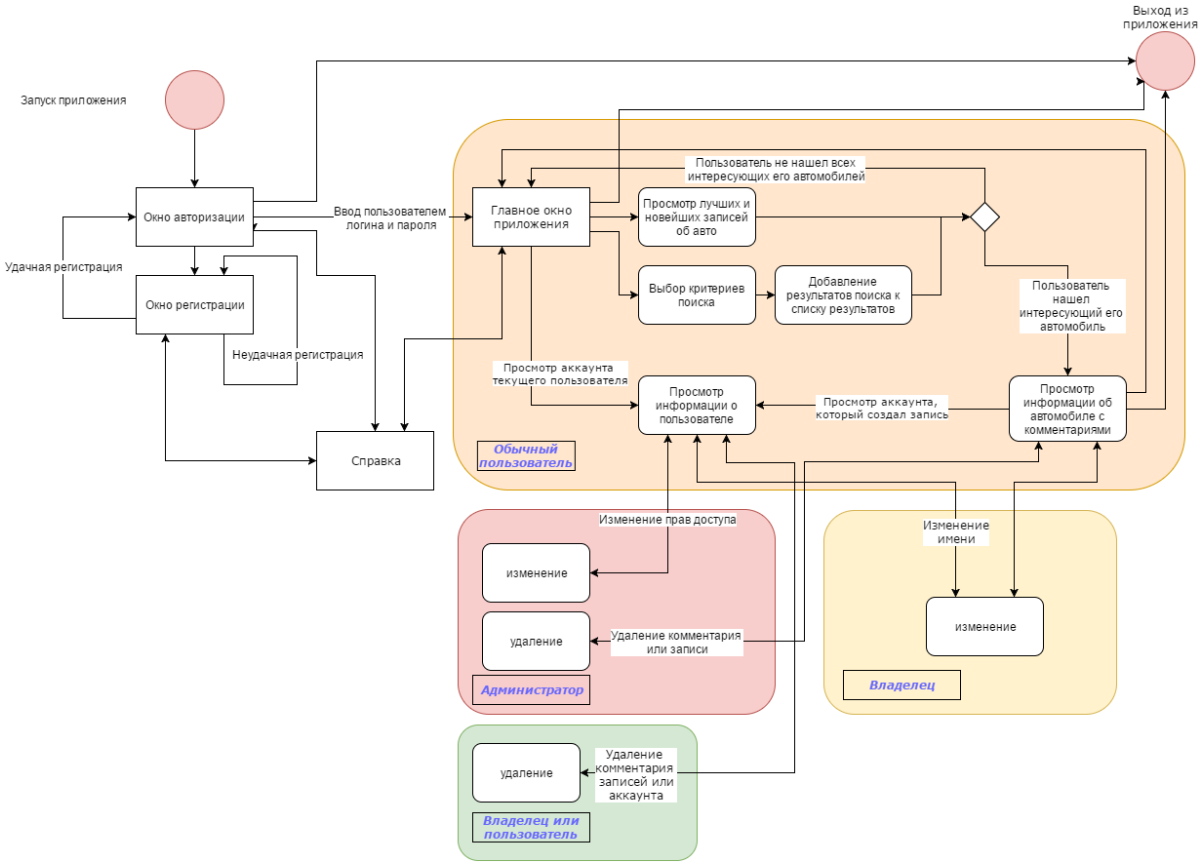


Рисунок 17

Начало и конец схемы обозначены красными кругами, действия пользователя обозначены в прямоугольниках. Стрелочки обозначают смену состояния или действие пользователя, текст около стрелок – уточнение того, как происходит переход.

4. Разработка автомобильного справочника

4.1 Создание пользовательского интерфейса

При создании интерфейса программы были использованы стандартные элементы Windows, стилизованные с помощью Windows Form Designer. Это значит, что большинство пользователей будут взаимодействовать со знакомыми элементами интерфейса, а это облегчает понимание пользователем функциональности программы. Также для облегчения понимания пользователем функционала в главную форму приложения был добавлен элемент ToolTip, который позволяет создавать графические подсказки, при наведении на некоторые элементы управления.

Сначала главное окно приложения было разбито на панели для разделения по функционалу, после чего к этим панелям были прикреплены элементы интерфейса. Для создания переключаемых окон внутри приложения был использован элемент TabControl, который был размещен так, что переключатели не были видны пользователю. Таким образом было сэкономлено время на создания пользовательского элемента, который бы проделывал эти же действия. Таким образом, интерфейс программы (главное окно и начальное окно просмотра новейших и лучших авто) обрел вид, как на рисунке 18.

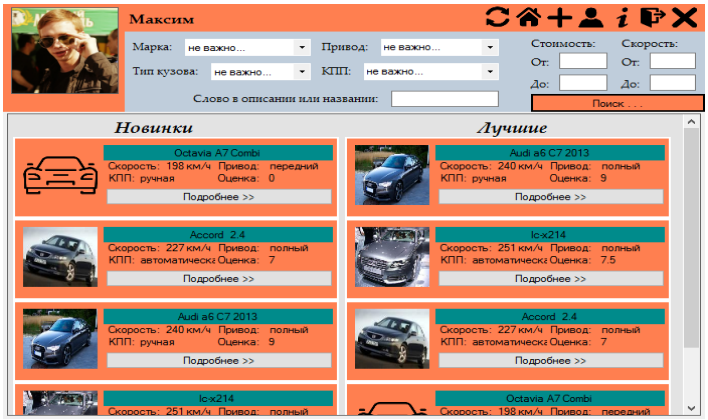


Рисунок 18

Все элементы ComboBox имеют вид DropList, это означает, что пользователь не может вбивать данные вручную, а может только выбирать из списка, аданного программно. На панели управления содержаться кнопки, стилизованные под популярные иконки, так что также прибавляет простоты в обращении со стороны пользователя. В элементе с фотографией пользователя содержится стандартная иконка пользователя. Данная иконка имеет на заднем фоне прозрачность, это позволит менять цвет задней панели, не меняя при этом стандартную картинку.

На рисунке 19 можно сравнить интерфейс главного приложения с открытой вкладкой просмотра информации об автомобиле, а также этот же вид, но в режиме редактирования записи.

Рисунок 19

Почти все элементы стали редактируемые, во всех Combo Box элементах нет права выбора своего варианта. Также возле фотографии появилась белая кнопка для вызова файлового диалога и последующего изменения фотографии автомобиля (рисунок 20).

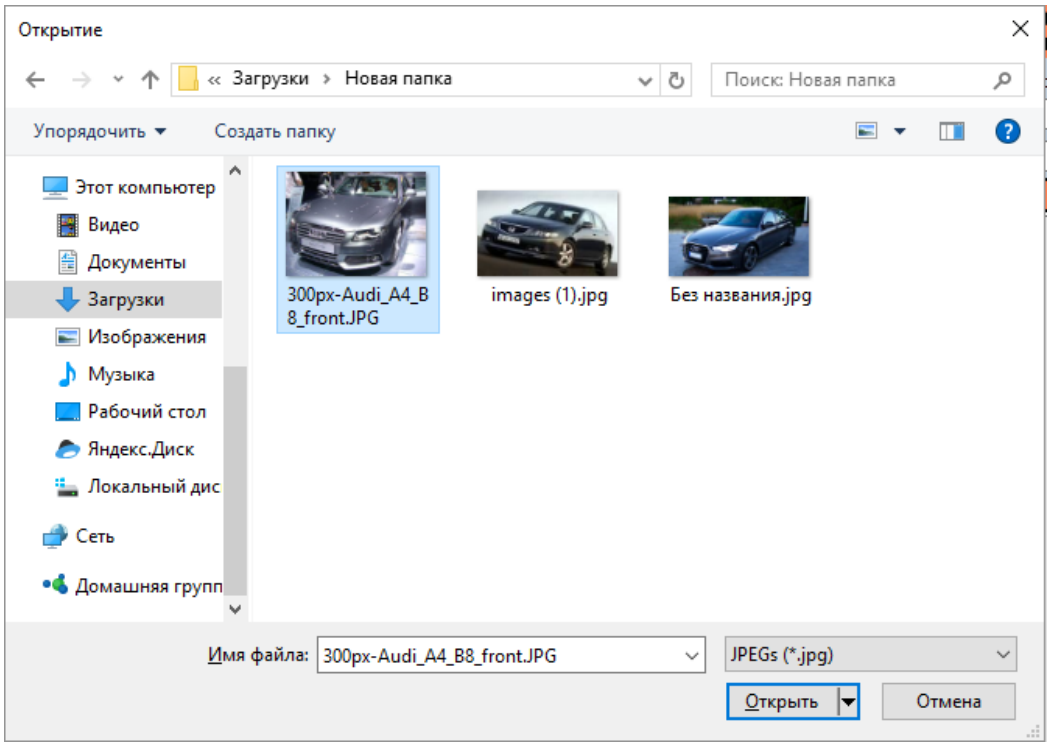


Рисунок 20

Как можно увидеть – в диалоге нельзя просматривать и выбирать любые файлы, а только изображения форматов: png, jpg, bmp, gif. В коде это было организованно следующим образом (рисунок 21):

```
1 reference
private void changeCarInfoImage_Click(object sender, EventArgs e)
{
    OpenFileDialog f = new OpenFileDialog();
    f.Filter = "PNG|*.png|JPEGs|*.jpg|Bitmaps|*.bmp|GIFs|*.gif";
    f.FilterIndex = 1;
    if (f.ShowDialog() == DialogResult.OK)
    {
        imageCI.Image = Image.FromFile(f.FileName);
    }
}
```

Рисунок 21

После отправления комментария, вид вкладки меняется и панель редактирования комментария исчезает, а на ее месте появляется последний комментарий пользователя (рисунок 22).

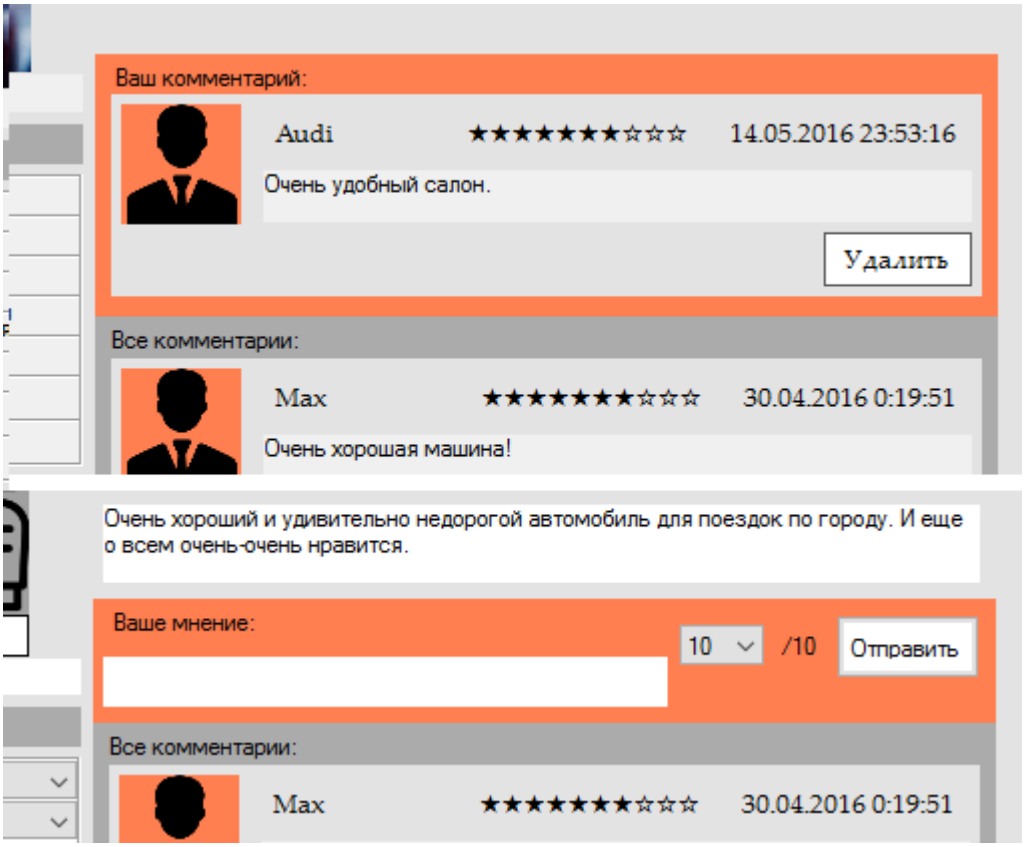


Рисунок 22

Вкладка просматривания и редактирования записей пользователя в разных режимах выглядит, как на рисунке 23. В режиме редактирования от модератора можно лишь изменять свое имя и фотографию, в то время, как администратор может изменить и чужие права. У каждого списка во вкладке личного кабинета есть кнопка «Еще», которая добавляет элементов соответствующие списки. Данная функция позволяет не загружать сразу все комментарии и записи пользователя, которых может быть большое множество.

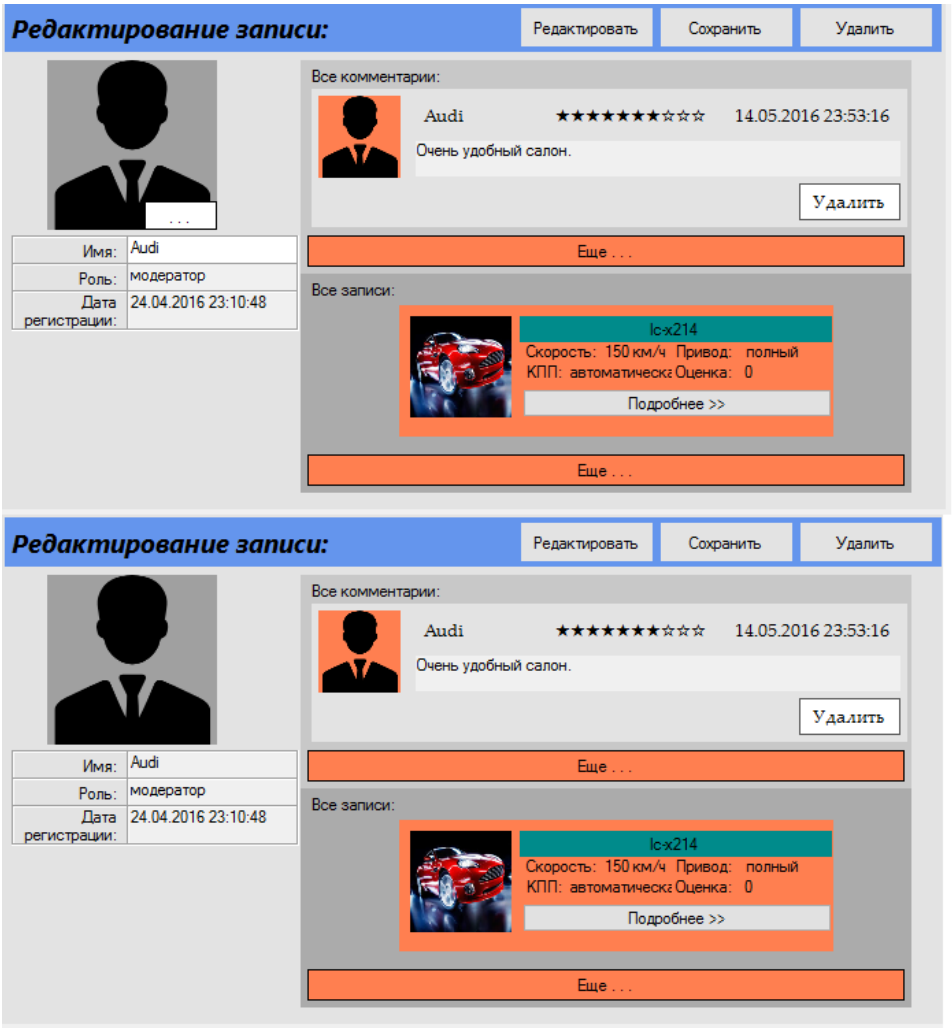


Рисунок 23

Чтобы попасть во вкладку результатов поиска следует:

1. Заполнить панель поиска (необязательно)
2. Нажать кнопку «Поиск...»

После чего откроется вкладка, интерфейс которой изображен на рисунке 24. Список панели поиска работает в соответствии с ТЗ, поэтому для того, чтобы очистить список следует нажать кнопку «очистить», а иначе результаты неоднократного поиска будут складываться в списке результатов.

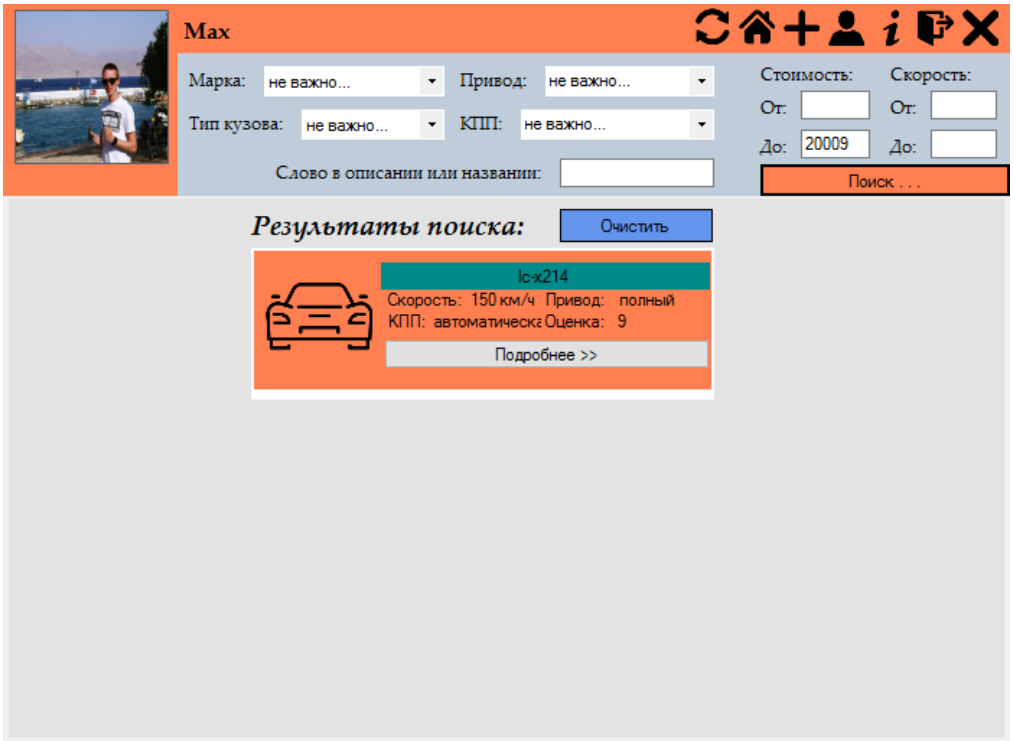


Рисунок 24

При запуске приложения или выходе из аккаунта, пользователя встречает окно входа в аккаунт (рисунок 25). Данная форма всего два поля ввода и четыре кнопки. Поля ввода нужны для ввода логина и пароля, а кнопки выполняют функционал, указанный в ТЗ.

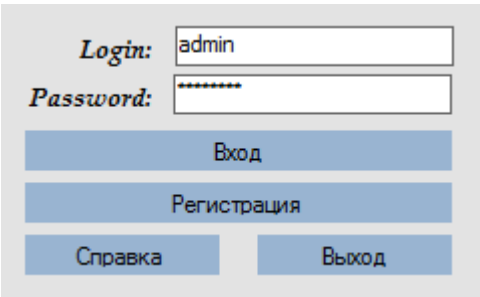


Рисунок 25

На форму регистрации (рисунок 26) можно попасть только из формы ввода. Данная форма имеет четыре поля ввода и три кнопки. По мимо отправки данных в базу, данная форма имеет метод проверки данных, который выдает ошибку, если:

- Одно из полей пустое

- Логин уже существует
- Пароли не совпадают
- Пароль меньше 8 знаков

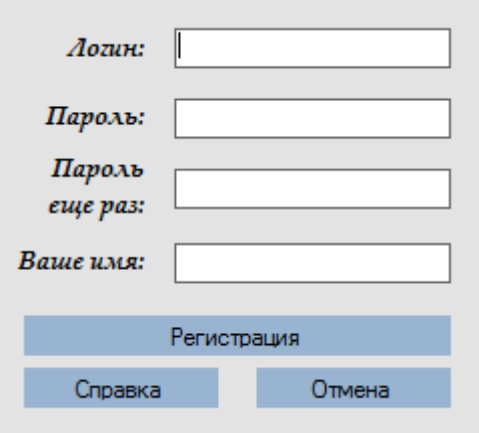


Рисунок 26

Из каждого окна приложения можно вызвать справку. Данная форма содержит три вкладки интерфейс которых можно изучить на рисунке 27. Вкладки организованы с помощью TabControl, а данные во вкладках «Введение» и «F.A.Q.» записи в виде списка, каждый элемент которого – панель, содержащая нужные элементы. Панели, содержащие данные записи имеют свойство AutoSize в значении true, что позволяет не редактировать размер панели при изменении внутренних элементов.

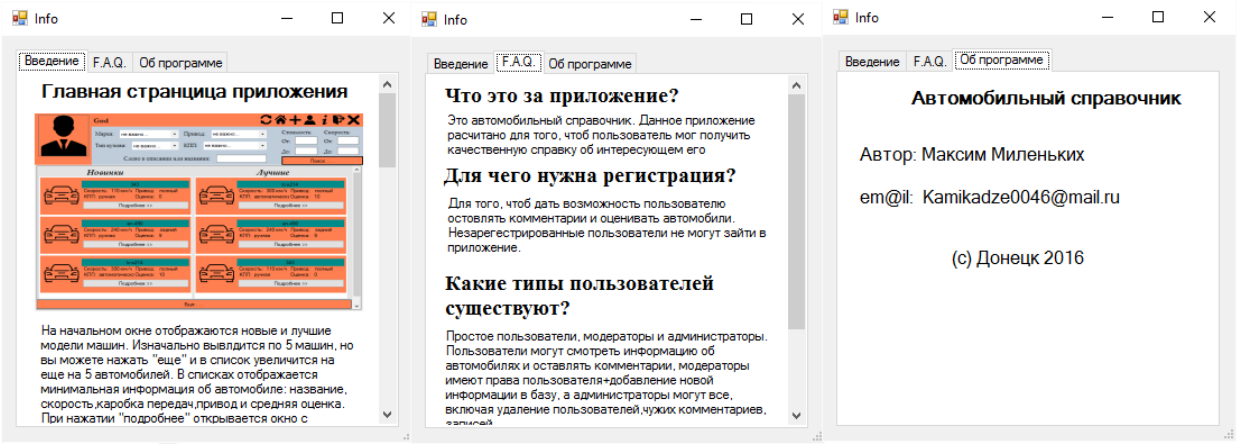


Рисунок 27

Одной из полезных возможностей Windows Form есть создание пользовательских элементов. Пользовательский элемент – это элемент, наследуемый от базового класса Control, которому можно самому задавать свойства и методы. Это становиться чрезвычайно полезным, когда в формах часто повторяются некоторые ее части или для некоторых элементов следует проделывать одни и те же действия. Так, например, в приложении был создан класс RRTB (Resizable Rich Text Box), который используется в других пользовательских элементах и его особенностью есть то, что при вводе текста данный элемент увеличивает и уменьшает размер в соответствии с количеством строк, в нем. Также в приложении присутствуют еще несколько пользовательских элементов:

Таблица 9 – Пользовательские елементы приложения и их методы

Название пользовательского элемента	Методы
RRTB	- RRTB(readonly, width) Создает элемент и устанавливает возможность его редактирования и максимальную ширину. - TextChanged При необходимости меняет размер элемента. - GetText Возвращает текст, содержащийся внутри элемента.
MinCarForm	- MinCarForm(CarInfo) Создание элемента на основе полученного класса, содержащего все данные об авто. - TakeMoreInfo Открывает вкладку просмотра и редактирования информации об авто в главном окне для автомобиля, переданного до этого в конструктор.

Окончание таблицы 9

CommentForm	-CommentForm(CommentClass, bool) Принимает объект класса комментария и булевскую переменную. Эти параметры устанавливают данные внутренних элементов класса, а также возможнось удаления элемента. -Remove () При нажатии на кнопку удаления данный элемент удаляет себя из списка родительского окна.

4.2 Разработка логики программы

Логикой программы можно назвать класс, методы и свойства которого выполняли функционал приложения, не привязанный к интерфейсу. Таким образом был создан класс App, который содержит локальные данные, экземпляр класса базы данных с подключенным соединением, а также некоторые методы, которые нужны для приложения. Данный класс статический, то есть содержится в одном экземпляре на все приложение и доступен во всех формах, что однозначно упрощает процесс разработки.

Таблица 10 - Методы, используемые в классе App

Названия метода	Реализация
App ()	Создает подключение к базе и пользователя.
bool login (string login, string password)	По логину и паролю пытается получить данные пользователя, если данная операция завершается успехом, то методом возвращается true, иначе - false.
Bool RefreshUserData()	Обновляет данные пользователя

Окончание таблицы 10

ReturnDriveType(int)	Возвращает строковое значение по значению типа привода. 0 – полный привод, 1 – задний, 2 – передний.
RatingToString(int)	Возвращает строку, в которой символов звезды столько, сколько указанно в передаваемом в метод параметре.
ReturnTransmission(int)	Возвращает строковое значение по значению коробки передач. 0 – автоматическая, 1 - ручная
ReturnRole(int)	Возвращает строковое значение по значению прав пользователя.

Но, не всю логику можно вынести в отдельный класс, так как большая часть функциональности привязана к элементам интерфейса. Поэтому были разработаны следящие методы в форма приложения:

Таблица 11 - Основные методы, используемые в классе MainForm

Названия метода	Реализация
RefreshMainForm	Основной метод обновления приложения, который содержит другие методы обновления. При запуске этого метода обновляются все нужные данные в БД, затем панель управления и главное окно, а затем выбранная вкладка на главном окне, при этом другие вкладки не обновляются, что укоряет работу приложения.

Окончание таблицы 11

RefreshCarInfo, RefreshUserInfo, RefreshBestAndNewestCars	Удаляет и очищает все динамические элементы управления, после чего заполняет их нужными данными.
SaveCarInfo_Click	Обновляет или создает запись об автомобиле.
SaveUserInfo_Click	Обновляет данные пользователя.
DelCarInfo_Click, DelUser_Click	Методы удаления записей об авто и пользователей.
AddCommentBtn_Click	Составляет и отправляет комментарий в БД.

4.3 Разработка взаимодействия с БД

Для того, чтоб сократить повторения в коде, а также облегчить читаемость и переносимость программы, как этого требовалось в ТЗ, требуется создать класс для работы с БД. У данного класса должны быть следующие атрибуты и методы:

- Экземпляр переменной, соединяемой с базой
- Экземпляр Строки подключения
- Методы работы с подключением
- Методы работы с данными приложения

Первые три нужны для гибкого использования класса, подключения (переподкючение, в случаях ошибок и т.д.), а последний пункт осуществляет связь между данными приложения и базы данных.

Проанализировав наборы данных, требуемых внутри приложения,

можно разбить методы работы с данными класса на следующие типы:

1. Методы получения одного объекта из базы
2. Методы получения списка объектов из базы
3. Методы удаления объекта в базе
4. Методы редактирования объекта в базе
5. Методы добавления объекта в базу
6. Метод проверки существования объекта

В таблице 12 содержатся методами класса DBManager.

Таблица 12 – методы класса DBManager

Тип метода	Методы	Описание
Управление соединением	bool isOpen(), bool Connect(), Void Close(), bool RefreshConnect().	Метод Connect и RefreshConnect возвращают true, если подключение прошло успешно.
Получения объекта из базы	UserClass ReturnUserById (int id), UserClass ReturnUserByLoginAndPassword(string login, string password), CarInfoClass ReturnCarInfoById(int id).	Получение объекта из базы по атрибутам
Получения списка объектов из базы	List<CommentClass> ReturnAllCommentsByAtr(string atr, int value), List<CarInfoClass> ReturnAllCarInfo(), List<string> ReturnAllCarMakeOrBodyType(string TableName,string atr).	Возвращает список классов. Последний метод работает сразу с двумя таблицами и принимает на вход имя таблицы и атрибут для возвращения.

Окончание таблицы 12

Удаления объекта в базе	DeleteTableById(string tableName, int id), DeleteCarInfo(int id), DeleteUserById(int id).	Удаляет все связанные объекты и сам объект из базы по идентификатору. DeleteTableById удаляет любую таблицу, которая ни с кем не связанна.
Редактирование объекта в базе	Bool ChangeCarInfo(CarinfoClass), Bool ChangeUserInfo(UserClass).	Меняют значение полей экземпляра таблицы в базе, на основании принятого параметра. (id меняемой таблицы в базе совпадает с id посланного в метод объекта).
Добавление объекта в базу	AddComment(CommentClass), AddUser(USerClass), AddCarInfo(CarInfoClass).	Все методы возвращают true, при успешном добавлении.

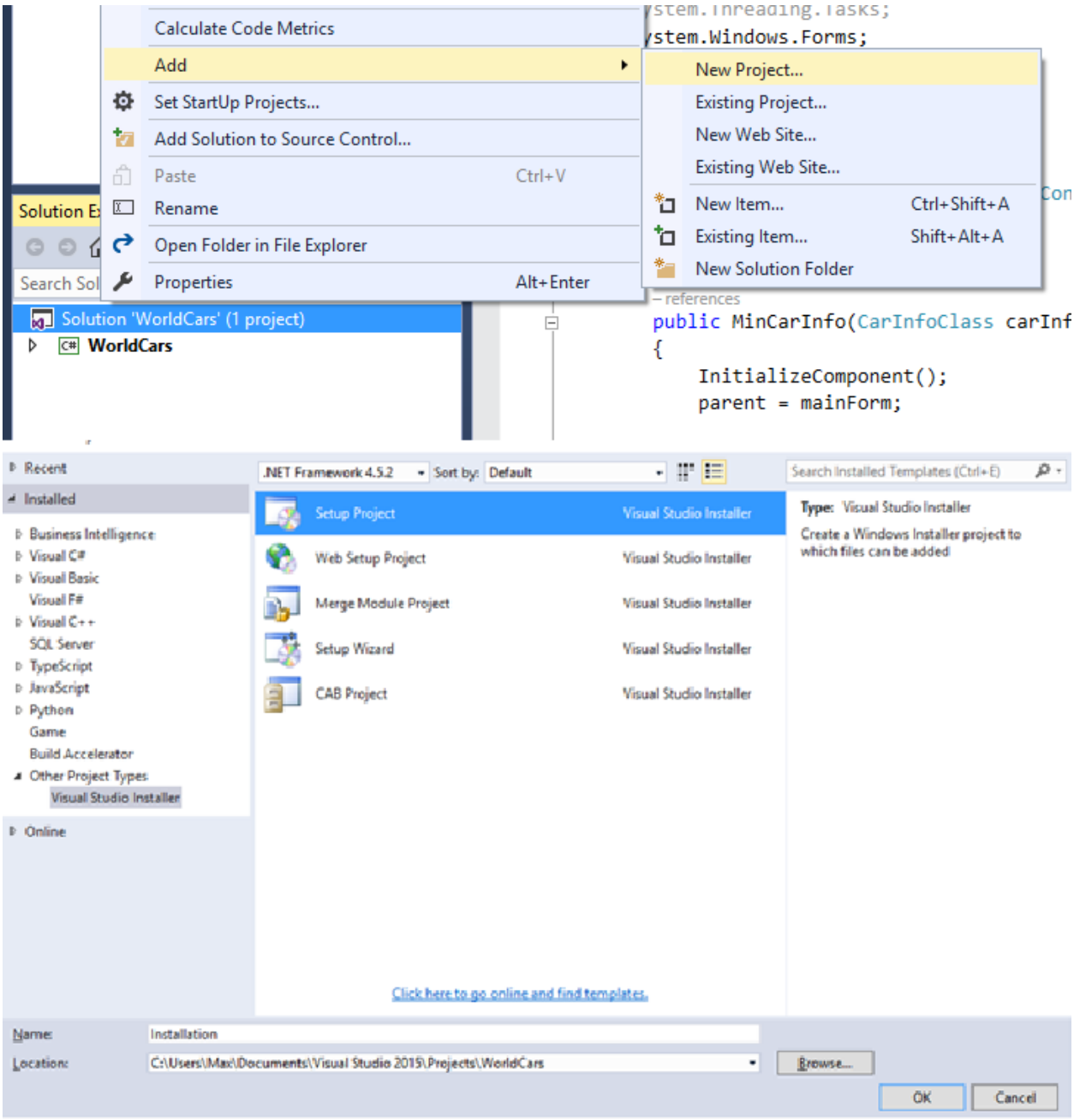
4.4 Создание инсталляционного файла

Для создания инсталляционного файла в данной КР используется инструмент Visual Studio Installer Projects, который позволяет создавать инсталляционные пакеты .msi для Windows Installer.

Windows Installer (установщик Windows) — подсистема Microsoft Windows, обеспечивающая установку программ. Является компонентом

Windows, начиная с Windows 2000; может устанавливаться и на более ранние версии Windows. Вся необходимая для установки информация (иногда и вместе с устанавливаемыми файлами) содержится в установочных пакетах (installation packages), имеющих расширение .msi. Файл .msi представляет собой составной документ OLE (OLE compound document — в том же формате-контейнере хранятся документы Microsoft Word, Excel и т. д.), в котором содержится небольшая реляционная база данных — набор из нескольких десятков взаимосвязанных таблиц, содержащих различную информацию о продукте и процессе установки. При этом все строковые данные в базе хранятся вместе в отдельном потоке документа, а в таблицах базы на них имеются ссылки; таким образом избегают дублирования строк, что значительно уменьшает размер базы. Кроме базы, структура файла .msi предусматривает помещение туда пользовательских сценариев и вспомогательных DLL, если таковые требуются для установки, а также самих устанавливаемых файлов, запакованных в формате .cab. Файлы можно размещать и отдельно от пакета, в запакованном или распакованном виде (с сохранением структуры каталогов).

После скачивания и установки инструмента Visual Studio Install Projects,в решении (на уровне проекта самого приложения) был создан проект Setup Project с названием Installation (рисунок 28). Далее, нужно открыть файловую систему проекта (installation-> view-> File system. . .) и добавить в него необходимые для работы приложения файлы. Для этого следует добавить Primary Output в папку Application Folder. Теперь, при построении приложения, скомпилированные файлы будут помещаться в установочный пакет. Однако, поскольку данное приложение еще работает с локальной базой, то нужно еще добавить ее в этот же пакет. Также стоит добавить некоторые ресурсы, которые не помещаются в проект при компиляции: два стандартных изображения пользователя и одно изображение автомобиля.


Calculate Code Metrics
Add
Set StartUp Projects...
Add Solution to Source Control...
Paste Ctrl+V
Rename
Open Folder in File Explorer
Properties Alt+Enter
New Project...
Existing Project...
New Web Site...
Existing Web Site...
New Item... Ctrl+Shift+A
Existing Item... Shift+Alt+A
New Solution Folder
Solution 'WorldCars' (1 project)
WorldCars
public MinCarInfo(CarInfoClass carInf
{
InitializeComponent();
parent = mainForm;

Recent
Installed
Business Intelligence
Visual C#
Visual Basic
Visual F#
Visual C++
SQL Server
TypeScript
JavaScript
Python
Game
Build Accelerator
Other Project Types
Visual Studio Installer
Online
.NET Framework 4.5.2
Sort by: Default
Search Installed Templates (Ctrl+E)
Setup Project Visual Studio Installer
Web Setup Project Visual Studio Installer
Merge Module Project Visual Studio Installer
Setup Wizard Visual Studio Installer
CAB Project Visual Studio Installer
Type: Visual Studio Installer
Create a Windows Installer project to which files can be added
Click here to go online and find templates.
Name: Installation
Location: C:\Users\Max\Documents\Visual Studio 2015\Projects\WorldCars
Browse...
OK
Cancel

Рисунок 28

5. Описание программного продукта

5.1 Структура проекта

На рисунке 29 изображена иерархическая структура проекта.

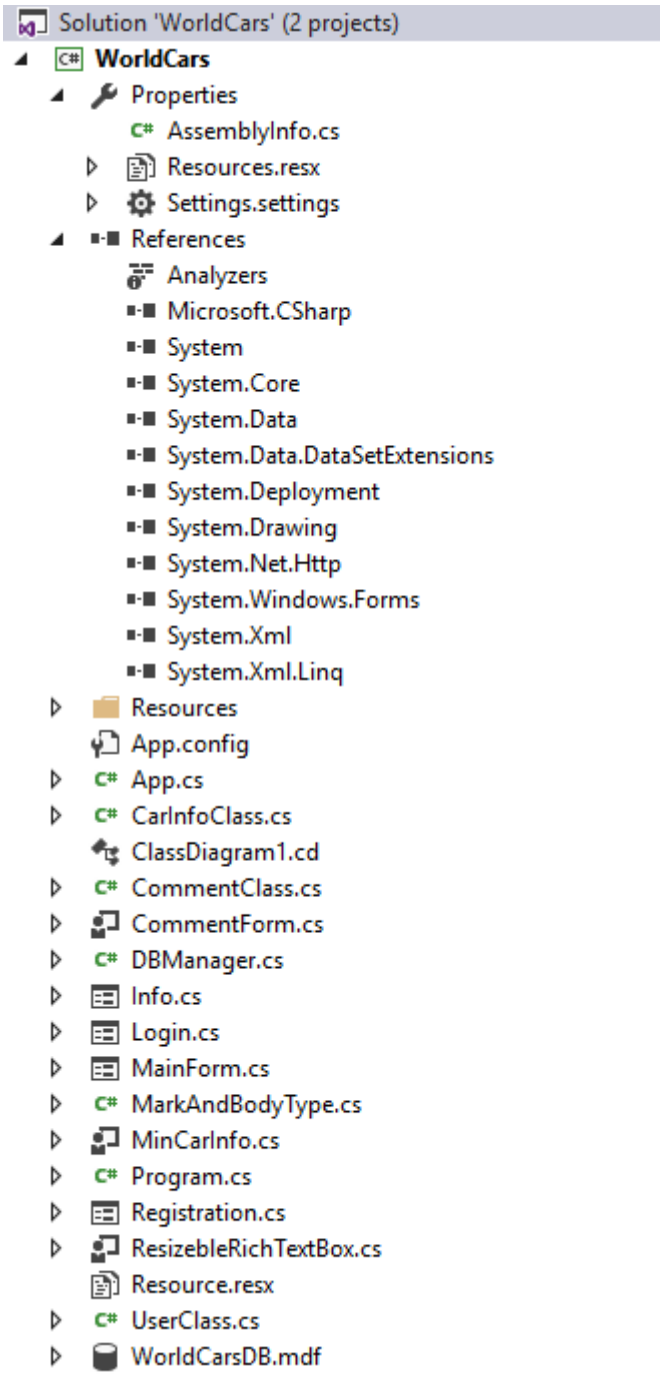


Рисунок 29

Рассмотрим детальнее компоненты проекта:

1. Properties содержит основные настройки приложения (целевая версия платформы .NET Framework, тип сборки и т.д).
   1.1. AssemblyInfo.cs – файл содержит в себе информацию о сборке.
   1.2. Resourses.resx – файл, описывающий ресурсы.
   1.3. Settings.settings – файл, в котором можно хранить некоторые настройки проекта для динамического изменения.
2. References - содержит все ссылки на внешние компоненты в проекте C#.
3. Resourses – папка с ресурсами проекта.
4. App.config – конфигурационный файл XML, который может содержать, например строку подключения к БД и т.д.
5. WorldCarsDB.mdf – локальная база данных проекта, созданная с помощью инструментов Visual Studio.

Далее идет множество созданных классов, группировку которых можно рассмотреть в таблице 13. Все пользовательские элементы интерфейса и формам привязаны файлы с такими же именами, но с расширениями .designer.cs и .resx. Последний нужен для хранения ресурсов, а первый для работы графического редактора Windows Form.

Таблица 13 – Классификация физических объектов проекта

Тип файла	Название файла(ов)	Описание
Пользовательский элемент интерфейса	CommentForm.cs, MinCarInfoForm.cs, ResizebleRichTexBox.cs.	Созданные элементы интерфейса, для упрощения создания интерфейса
Формы приложения	MainForm.cs, Login.cs, Info.cs, Registration.cs.	Формы, содержащие весь интерфейс приложения.
Пользовательский класс	DBManager.cs	Класс, взаимодействующий с БД.
	App.cs.	Класс с основной логикой и локальными данными приложения
	CommentClass, UserClass, CarInfoClass, MarkAndBodyType.	Классы, описывающие сущности приложения.

Помимо проекта приложения, есть еще проект инсталлятора, который находится на том же уровне (в решении). Данный проект намного меньше, но имеет совершенно иную иерархическую структуру (рисунок 30).

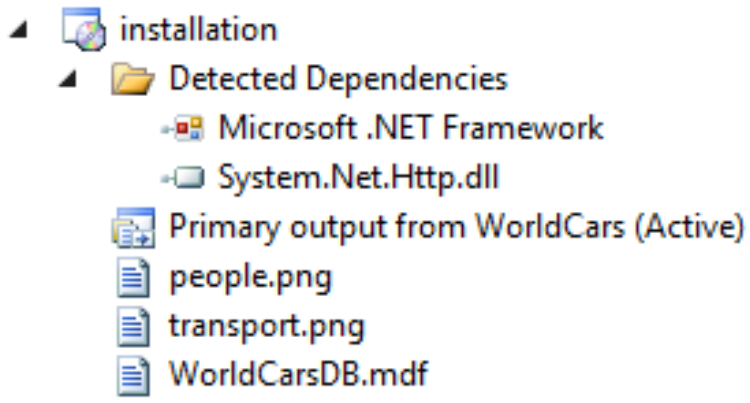


Рисунок 30

Данный проект installation содержит:

1. Список зависимостей проекта приложения.
2. Primary Output – основные выходные при компиляции файлы.
3. Дополнительные файлы для распаковки в папку программы: база данных, стандартные картинки.

5.2 Описание объектов и их взаимодействия

На рисунке 31 взаимодействие объектов может показаться сложным, но на самом деле это не так. При запуске приложения запускается класс Program и его метод main, в котором создается объект App и форма LoginForm. При нажатии кнопки «Вход» открывается диалоговое окно MainForm, в котором и происходит основная работа приложения. Если пользователь хочет выйти из приложения совсем, то форма закрывается с возвратом результата DialogResult.

Класс App, отвечающий за логику, содержит локальные данные и нужные методы для приложения, таким образом он связан с классами UserClass и DBManage.

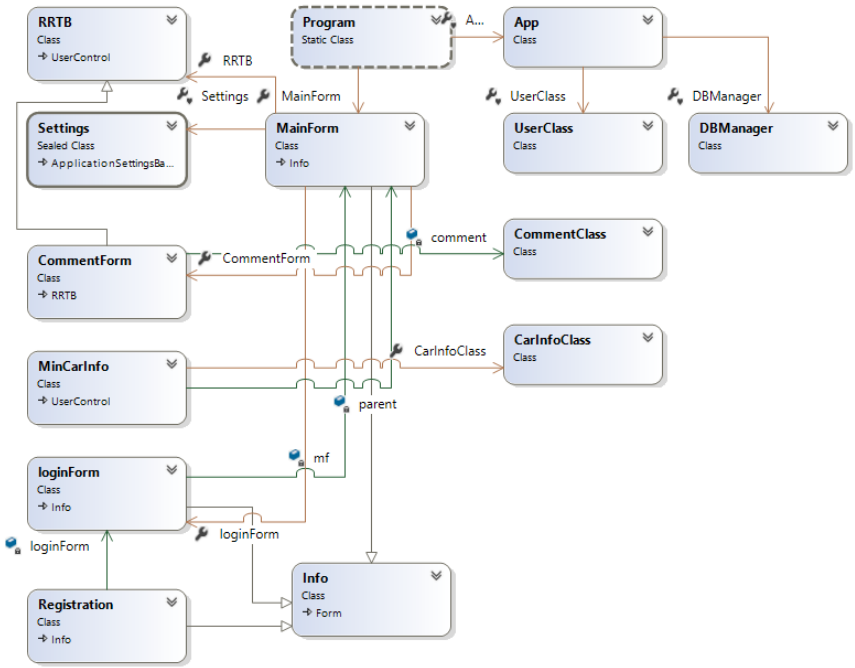


Рисунок 31

MainForm физически не связан с какими-либо классами сущностей приложения, но содержит переменные, в которых находятся идентификаторы объектов в БД. Поскольку, в mainForm сосредоточен почти весь интерфейс приложения, то внутри этой формы находятся все виды пользовательских элементов, а также из этой формы можно вызвать формы входа в аккаунт и справку.

Форма Login – это единственная форма, которая может вызвать форму Registration. Обе формы имеют небольшой функционал в виде валидации введенной информации и связи с БД через класс Program. Также данные формы могут вызвать форму справки.

Остальные классы, это классы, представляющие сущности приложения. Это классы: CommentClass, UserClass, CarInfoClass, MarkAndBodyType. Все они так или иначе используются в приложении. Например, когда какой-либо метод MainForm через класс App вызывает метод класса DBWorker, то, в основном, принимаемыми и передаваемыми данными являются эти самые объекты.

5.3 Описание SQL-запросов

Все работа с базой данных, а соответственно и sql-запросы, находится в классе DBManager. В таблице 14 показаны методы приложения, а также основную часть их запроса к БД.

Таблица 14 – Связь DBManager с базой

Тип метода	Методы	Основное содержание sql-команды
получение объекта из базы	ReturnUserById	Select * from [имя таблицы]
	UserClass	
	ReturnUserByLoginAndPassword	
	ReturnCarInfoById	
	ReturnAllCommentsByAtr	
	ReturnAllCarInfo	
	ReturnAllCarMakeOrBodyType	
удаления объекта в базе	DeleteTableById	Delete from [имя таблицы]
	DeleteCarInfo	
	DeleteUserById	
Редактирование объекта в базе	ChangeCarInfo	Update [имя таблицы] set (список из «атрибут=параметр»)
	ChangeUserInfo	

Окончание таблицы 14

Добавление объекта в базу	AddComment	Insert into [имя таблицы] (список атрибутов таблицы) Value(список передаваемых параметров)
	AddUser	
	AddCarInfo	

В методах обновление таблиц (список из «атрибут=параметр») означает, что после слова «set» идет список, элементами которого являются выражения присваивания, где атрибуту таблицы приравнивается значение параметра, передаваемого в базу. Значение параметра устанавливается с помощью класса SqlCommand (рисунок 32), с помощью которого создаются и посылаются команды в базу. Команда вместе с открытым подключением посылаются в конструктор этого класса, а затем выполняется с помощью одного из следующих методов: ExecuteReader или ExecuteNonQuery. Первый возвращает объект, с помощью которого производится считывание данных из БД, а второй возвращает количество элементов БД, затронутых выполнимой командой.

```
// Добавить параметры
cmd.Parameters.AddWithValue("@body_type", carInfo.body_type);
cmd.Parameters.AddWithValue("@car_make", carInfo.car_make);
cmd.Parameters.AddWithValue("@promoter_id", carInfo.promoter_id);
cmd.Parameters.AddWithValue("@description", carInfo.description);
cmd.Parameters.AddWithValue("@max_speed", carInfo.max_speed);
cmd.Parameters.AddWithValue("@cost", carInfo.cost);
cmd.Parameters.AddWithValue("@rating", carInfo.rating);
```

Рисунок 32

Также, метод AddCarInfo содержит команду «select @@IDENTITY», которая возвращает id последнего добавленного экземпляра таблицы CarInfo

.

6. Тестирование и внедрение

6.1 Тестирование разработанного ПО

Назначение, область применения и требования к надежности программного продукта показаны в таблице 15, а результаты анализа надежности в таблице 16.

Таблица 15 – Назначение, область применения и требования к надежности

Назначение	Область применения	Требования к надежности
Предоставление справки об автомобилях	Поиск информации	Ограничение доступа непроверенных пользователей

Таблица 16 - Анализ надежности программного продукта

№ Теста	Данные для теста (вводимая информация)	Время функционирования программы	Результат (успешно \отказ)	Результат
1.	Запуск приложения	5 секунд	успешно	Открытие формы входа в аккаунт
2.	Ввод корректных данных для входа	Меньше секунды	успешно	Авторизация в приложении
3.	Ввод некорректных данных для входа	Моменталь но	успешно	Предупреждение о неверном вводе данных

Продолжение таблицы 16

4.	Ввод корректных данных для регистрации	Меньше секунды	успешно	Регистрация в базе, предупреждение перенаправление пользователя к окну входа.
5.	Ввод некорректных данных для регистрации	Моментально	успешно	Предупреждение о неверном вводе данных
6.	Вызов функции создания записи об автомобиле с требуемыми правами доступа	Моментально	успешно	Открытие вкладки редактирования записи в режиме создания
7.	Вызов функции создания записи об автомобиле с требуемыми правами доступа	Моментально	успешно	Предупреждение о том, что не хватает прав
8.	Вызов функции сохранения записи об автомобиле с некорректно введенными данными	Моментально	успешно	Предупреждение о неверном вводе данных

Окончание таблицы 16

9.	Удаление собственного аккаунта	Меньше секунды	успешно	Перенаправление пользователя в окно входа в аккаунт
10	Отправка комментария	Меньше секунды	успешно	Добавление комментария в базу и обновление пользовательского интерфейса
11	Удаление комментария	Меньше секунды	успешно	Удаление комментария из базы и обновление пользовательского интерфейса

6.2 Установка программы

Для установки и использования разрабатываемого автомобильного справочника действуют следующие системные и программные требования:

- Не менее 300 kb памяти на жестком диске
- Операционная система Windows XP/ Windows7/8/10
- Процессор – Intel Pentium CPU 2117U@ 1.80GHz (2 ядра).
- Видеокарта от 256 mb памяти
- Оперативная память от 1 GB и выше

Также, для разработки данной программы нужны следующие системные и программные требования:

- Не менее 100 mb памяти на жестком диске (проект содержит локальную базу данных, поэтому чем больше места – тем лучше)
- Требуется установленная SqlLocalDB
- Операционная система Windows XP/ Windows7/8/10
- .NET framework версии 4.5.2
- Процессор – Intel Pentium CPU 2117U@ 1.80GHz (2 ядра).
- Видеокарта от 256 mb памяти
- Оперативная память от 1 GB и выше

1. Запустите файл installation.msi от имени администратора, нажмите «Далее» в появившемся окне диалога (рисунок 33).

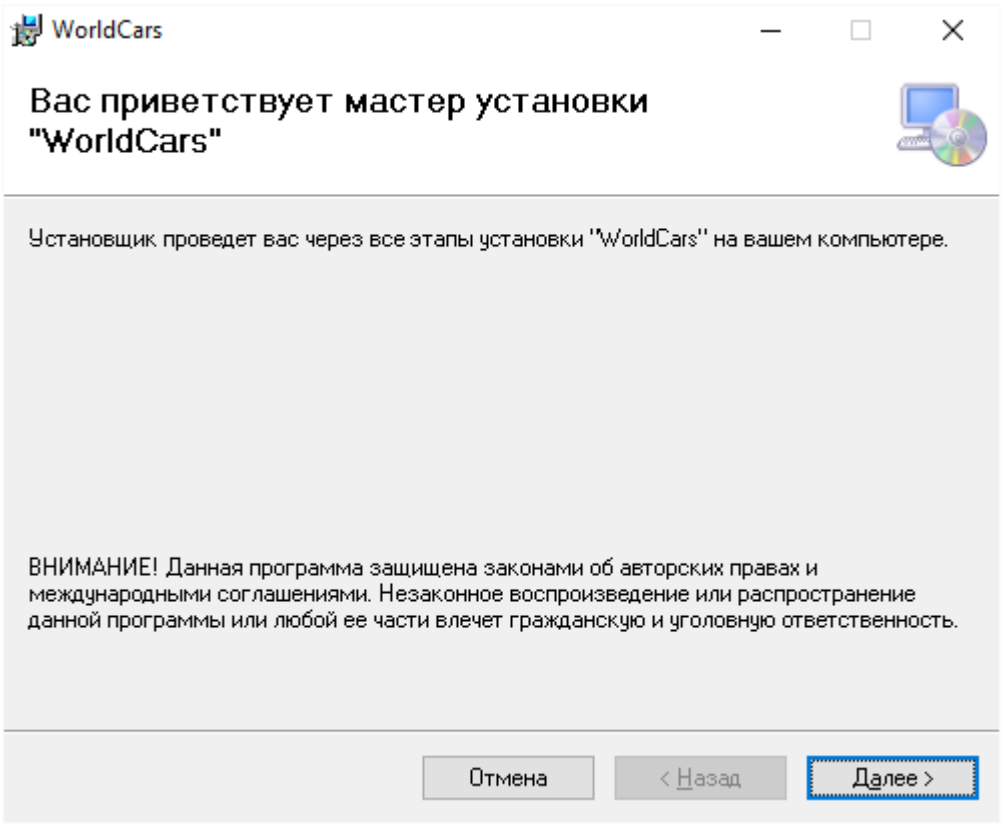


Рисунок 33

2. Выберите папку для установки и пользователя, для которого установите программу (рисунок 34) и нажмите «далее».

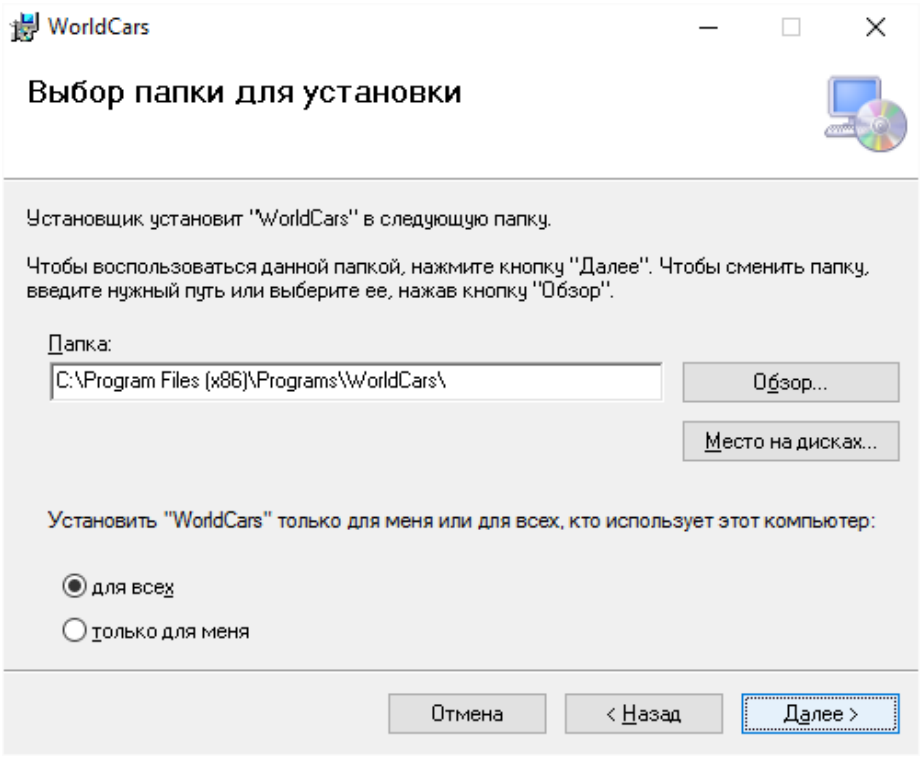


Рисунок 34

3. Нажмите «далее» в подтверждении установки (рисунок 35)

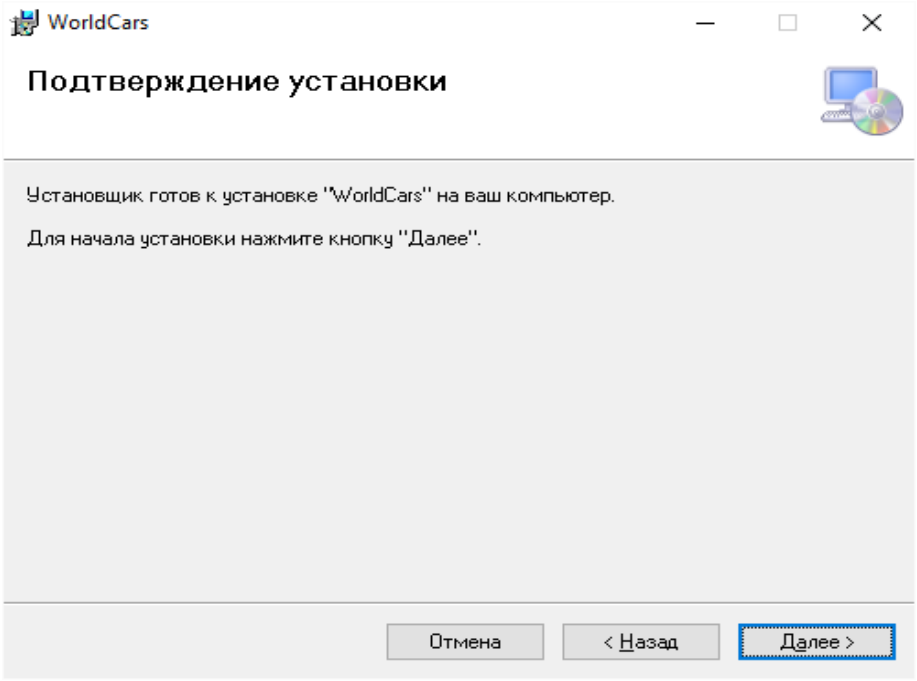


Рисунок 35

4. Дождитесь установки программы и подтвердите завершение установки (рисунок 36).

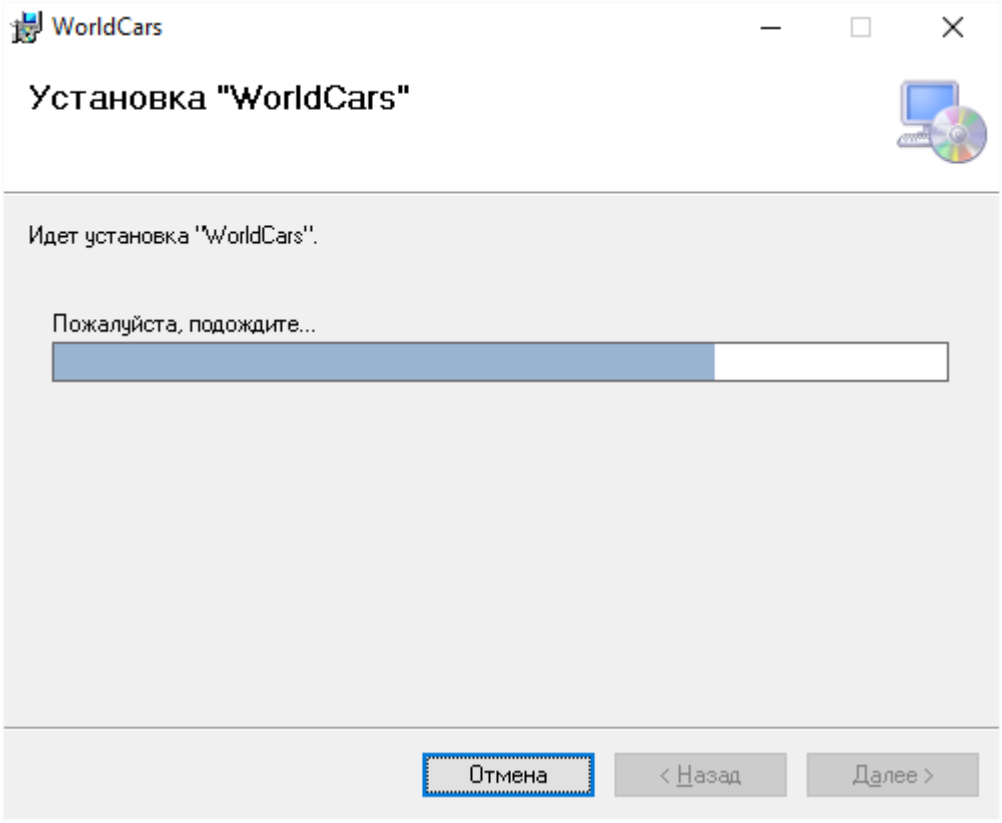


Рисунок 36

ЗАКЛЮЧЕНИЕ

В результате работы над курсовой работой выполнены все поставленные задачи. Разработано работоспособное приложение для информирования пользователей об разнообразных моделях автотранспорта. Проанализированы существующие аналоги, выделены базовые критерии оценки. Был изучен ряд технологий для создания проекта: C#, .NET, Windows Form, ADO.NET, MS SQL, Visual Studio 2015, Visual Studio Project Install. Также было написано обоснование выбора данных технологий в курсовом проекте. Функционал приложения был утвержден согласно разработанной структуре программы и базы данных. На основе разработанной функциональной схемы проекта составлен программный продукт, обладающий интерфейсной частью и базой данных для хранения данных об автомобилях, пользователях, комментариях, марках и типах кузова автомобилей.

На данный момент при работе с приложением доступны следующие возможности: регистрация, авторизация пользователей, просмотр профилей пользователей, возможность изменения личных данных, возможности администрирования, возможность создания/редактирования/удаления записей об автомобилях, а также возможность оценить записи пользователей.

Дальнейшее развитие программы связано с расширением ее возможностей, улучшением уровня безопасности, подключением к удаленной базе данных. Тестирование программы показало ее работоспособность и высокую степень надежности.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Официаоьный сайт BMW в России

URL: http://www.bmw.ru/ru/ (дата обращения: 25.04.2016)

2. Автомобильный форум AboutCar

URL: http://aboutcar.ru/ (дата обращения: 25.04.2016)

3. Wikipedia – информационный ресурс:

URL: https://ru.wikipedia.org/wiki/Windows_Forms

(дата обращения: 25.04.2016)

URL: https://ru.wikipedia.org/wiki/.NET_Framework

(дата обращения: 02.05.2016)

URL: https://ru.wikipedia.org/wiki/Microsoft_Visual_Studio

(дата обращения: 02.05.2016)

URL: https://ru.wikipedia.org/wiki/Установщик _Windows

(дата обращения: 04.05.2016)

4. Professional C# 5.0 and .NET 4.5 / Нейгел К., Ивьен Б. / М.: Вильямс, Год: 2014

5. Dependency Injection in .NET Букинистическое издание / Марк Симан, А. Барышнев, Евгений Зазноба / Питер

5. MSDN: информационный ресурс

URL:

(дата обращения: 12. 04.2016)

7. Esate.ru

URL: http://esate.ru/uroki/OpenGL/uroki-OpenGL-c-sharp/osnovi-windows-forms/

(дата обращения: 13. 04.2016)

8. Professorweb.ru // [ресурс с материалами для работы с ADO.NET]//

URL: http://professorweb.ru/my/ADO_NET/base/level1/ado_net_index.php (дата посещения: 22.04.2016)

ПРИЛОЖЕНИЕ А

(обязательное)

Фрагменты листинга

Листинг А.1 Класс DBManager

```
namespace WorldCars
{
    class DBManager
    {
        SqlConnectionStringBuilder connectStringBuilder;
        public SqlConnection connection;

        public DBManager()
        {
            connectStringBuilder = new
SqlConnectionStringBuilder();
            connectStringBuilder.DataSource =
@"(localdb)\MSSQLLocalDB";
            connectStringBuilder.ConnectTimeout = 30;
            connectStringBuilder.IntegratedSecurity = true;
            connectStringBuilder.AttachDBFilename =
@"|DataDirectory|WorldCarsDB.mdf";

            connection = new SqlConnection();
            connection.ConnectionString =
connectStringBuilder.ConnectionString;
        }
        public bool Connect()
        {
            try
            {
                //Открыть подключение
                connection.Open();
```

```
                return true;
            }
            catch (SqlException ex)
            {
                // Протоколировать исключение
                Console.WriteLine(ex.Message);
                return false;
            }
        }
        public void Close()
        {
            connection.Close();
        }
        public bool IsOpen()
        {
            return connection.State ==
System.Data.ConnectionState.Open;
        }
        public bool RefreshConnect()
        {
            Close();
            return Connect();
        }


        public List<CommentClass> ReturnAllCommentsByAtr(string
atribute,int value)
        {
            List<CommentClass> result = new
List<CommentClass>();

            string command = string.Format("SELECT * FROM
[Comment] WHERE {0}='{1}'", atribute, value.ToString());

            bool allFine = false;
```

```
            try
            {
                SqlCommand cmd = new SqlCommand(command,
connection);
                SqlDataReader dr = cmd.ExecuteReader();
                while (dr.Read())
                {
                    CommentClass buf = new CommentClass();
                    buf.id = (int)dr[dr.GetOrdinal("Id")];
                    buf.user_id =
(int)dr[dr.GetOrdinal("user_id")];
                    buf.rating =
(int)((float)dr[dr.GetOrdinal("rating")]);
                    buf.text =
(string)dr[dr.GetOrdinal("text")];
                    buf.datetime =
(DateTime)dr[dr.GetOrdinal("datetime")];

                    result.Add(buf);
                }
                dr.Close();
                allFine = true;
            }
            catch (SqlException ex)
            {
                // Протоколировать исключение
                Console.WriteLine(ex.Message);
            }


            if (allFine)
                return result;
            else
                return null;
```

```
        }
        public List<CarInfoClass> ReturnAllCarInfo(string
command)
        {
            List<CarInfoClass> result = new
List<CarInfoClass>();

            bool allFine = false;
            try
            {
                SqlCommand cmd = new SqlCommand(command,
connection);
                SqlDataReader dr = cmd.ExecuteReader();
                while (dr.Read())
                {
                    CarInfoClass buf = new CarInfoClass();
                    buf.id = (int)dr[dr.GetOrdinal("Id")];
                    buf.body_type =
(string)dr[dr.GetOrdinal("body_type")];
                    buf.car_make =
(string)dr[dr.GetOrdinal("car_make")];
                    buf.promoter_id =
(int)dr[dr.GetOrdinal("promoter_id")];
                    if
(dr.IsDBNull(dr.GetOrdinal("description"))) buf.description =
"";
                    else  buf.description =
(string)dr[dr.GetOrdinal("description")];

                    buf.max_speed =
(int)dr[dr.GetOrdinal("max_speed")];

                    buf.cost = (int)dr[dr.GetOrdinal("cost")];
                    buf.datetime =
(DateTime)dr[dr.GetOrdinal("datetime")];
```

```
                    if (dr.IsDBNull(dr.GetOrdinal("rating")))
buf.rating = 0;
                    else buf.rating =
(float)dr[dr.GetOrdinal("rating")];

                    buf.drive_type =
(int)dr[dr.GetOrdinal("drive_type")];

                    buf.transmission =
(int)dr[dr.GetOrdinal("transmission")];

                    buf.name =
(string)dr[dr.GetOrdinal("name")];

                    if (dr.IsDBNull(dr.GetOrdinal("image")))
buf.image = null;
                    else  buf.image =
(byte[])dr[dr.GetOrdinal("image")];

                    result.Add(buf);
                }
                dr.Close();
                allFine = true;
            }
            catch (SqlException ex)
            {
                // Протоколировать исключение
                Console.WriteLine(ex.Message);
            }
            if (allFine)
                return result;
            else
                return null;
```

```
        }
        public List<string> ReturnAllMarkOrBodyType(string
table,string atr)
        {
            List<string> result = new List<string>();
            string command = string.Format("select * from
[{0}]",table);
            bool allFine = false;
            try
            {
                SqlCommand cmd = new SqlCommand(command,
connection);
                SqlDataReader dr = cmd.ExecuteReader();
                while (dr.Read())
                {
                    string buf;
                    buf = (string)dr[dr.GetOrdinal(atr)];

                    result.Add(buf);
                }
                dr.Close();
                allFine = true;
            }
            catch (SqlException ex)
            {
                // Протоколировать исключение
                Console.WriteLine(ex.Message);
            }
            if (allFine)
                return result;
            else
                return null;

        }
```

```
        public UserClass ReturnUserById(int promoter_id)
        {
            UserClass result = new UserClass();
            string command = string.Format("SELECT * FROM [User]
WHERE id ='{0}'", promoter_id);

            bool allFine = false;
            try
            {
                SqlCommand cmd = new SqlCommand(command,
connection);
                SqlDataReader dr = cmd.ExecuteReader();
                if (dr.Read())
                {
                    result.id = (int)dr[dr.GetOrdinal("Id")];
                    result.login =
(string)dr[dr.GetOrdinal("login")];
                    result.password =
(string)dr[dr.GetOrdinal("password")];
                    result.access_level =
(int)dr[dr.GetOrdinal("access_level")];
                    result.name =
(string)dr[dr.GetOrdinal("name")];
                    result.datetime =
(DateTime)dr[dr.GetOrdinal("datetime")];
                    if (dr.IsDBNull(dr.GetOrdinal("image")))
result.image = null;
                    else result.image =
(byte[])dr[dr.GetOrdinal("image")];

                    allFine = true;

                }
                dr.Close();
            }
```

```
            catch (SqlException ex)
            {
                // Протоколировать исключение
                Console.WriteLine(ex.Message);
            }
            if (allFine)
                return result;
            else
                return null;
        }
        public UserClass ReturnUserByLoginAndPassword(string
login,string password)
        {
            UserClass result = new UserClass();
            string command = string.Format("SELECT * FROM [User]
WHERE login ='{0}' and password='{1}'", login,password);

            bool allFine = false;
            try
            {
                SqlCommand cmd = new SqlCommand(command,
connection);
                SqlDataReader dr = cmd.ExecuteReader();
                if (dr.Read())
                {
                    result.id = (int)dr[dr.GetOrdinal("Id")];
                    result.login =
(string)dr[dr.GetOrdinal("login")];
                    result.password =
(string)dr[dr.GetOrdinal("password")];
                    result.access_level =
(int)dr[dr.GetOrdinal("access_level")];
                    result.name =
(string)dr[dr.GetOrdinal("name")];
```

```
                    result.datetime =
(DateTime)dr[dr.GetOrdinal("datetime")];
                    if (dr.IsDBNull(dr.GetOrdinal("image")))
result.image = null;
                    else result.image =
(byte[])dr[dr.GetOrdinal("image")];
                    allFine = true;

                }
                dr.Close();
            }
            catch (SqlException ex)
            {
                // Протоколировать исключение
                Console.WriteLine(ex.Message);
            }
            if (allFine)
                return result;
            else
                return null;
        }
        public bool UserExist(string login)
        {
            bool result = false;
            string command = string.Format("SELECT * FROM [User]
WHERE login ='{0}'", login);
            try
            {
                SqlCommand cmd = new SqlCommand(command,
connection);
                SqlDataReader dr = cmd.ExecuteReader();
```

Листинг А.2 - Класс MainForm.cs.

```
    public partial class MinCarInfo : UserControl
```

```
    {
        MainForm parent;
        int car_id;
        public MinCarInfo(CarInfoClass carInfo,MainForm
mainForm)
        {
            InitializeComponent();
            parent = mainForm;
            if (carInfo.image == null)
            {
                pictureBox.Image =
Image.FromFile("transport.png");
            }
            else
            {
                MemoryStream ms = new
MemoryStream(carInfo.image);
                pictureBox.Image = Image.FromStream(ms);
            }
            maxSpeedLbl.Text = carInfo.max_speed.ToString() + "
км/ч";
            carNameLbl.Text = carInfo.name.ToString();
            transmissionLbl.Text =
App.ReturnTransmission(carInfo.transmission);
            driveTypeLbl.Text =
App.returnDriveType(carInfo.drive_type);
            ratingLbl.Text = carInfo.rating.ToString();
            car_id = carInfo.id;
        }
        private void takeMoreInfoBtn_Click(object sender,
EventArgs e)
        {
            parent.OpenCarInfo(car_id);
        }}
```