Федеральное государственное бюджетное образовательное учреждение высшего образования

**«ФИНАНСОВЫЙ УНИВЕРСИТЕТ ПРИ ПРАВИТЕЛЬСТВЕ**

**РОССИЙСКОЙ ФЕДЕРАЦИИ»**

**Департамент анализа данных,**

**принятия решений и финансовых технологий**

**Пояснительная записка к междисциплинарной курсовой работе**

на тему:

**Разработка информационно-справочной системы**

**«Прокат автомобилей»**

Выполнил:

студент группы ПИ2-2

Номоконов Иван Александрович

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Подпись)

Научный руководитель:

доцент, к.т.н. Горелов Сергей Витальевич

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Подпись)

2019

**Оглавление**

[Введение 2](#_Toc6013416)

[1. Постановка задачи 2](#_Toc6013417)

[2. Описание предметной области 3](#_Toc6013418)

[3. Актуальность автоматизации 4](#_Toc6013419)

[4. Описание программы 4](#_Toc6013420)

[4.1. Алгоритмические решения 4](#_Toc6013421)

[4.2. Описание интерфейса программы 6](#_Toc6013422)

[4.3. Состав приложения 14](#_Toc6013423)

[5. Назначение и состав классов программы 14](#_Toc6013424)

[5.1. Диаграмма классов 14](#_Toc6013425)

[5.2. Классы приложения 15](#_Toc6013426)

[5.3. Служебные классы 16](#_Toc6013427)

[5.4. Формы 17](#_Toc6013428)

[Заключение 18](#_Toc6013429)

[Список литературы 18](#_Toc6013430)

[Приложение. Исходный код программы 18](#_Toc6013431)

# Введение

Целью курсовой работы является приобретение навыков в анализе предметной области и в разработке сложного Windows-приложения на языке программирования C# в среде Visual Studio. Язык С# является достаточно мощным языком для разработки приложений для ПК. Разработка приложения позволит усовершенствовать знания этого языка, а также методов проектирования и анализа предметной области.

Анализ предметной области позволяет разработать информационную модель данных задачи. Во время анализа производится выбор наиболее важных объектов из предметной области и процессов автоматизации, а также разработка новых подходов и функций для решения проблем предметной области.

Также важной целью курсовой работы является разработка пояснительной записки, отвечающей таким требованиям, как полнота, лаконичность, техническая корректность и т.д. Техническая документация является составной частью программного продукта и, следовательно, определяет его качество. Поэтому приобретение навыков в разработке технической документации, а в данном случае пояснительной записки, также является важной задачей.

# 1. Постановка задачи

1. В курсовом проекте должна быть разработана информационная модель предметной области, представленная в виде пользовательских классов или таблиц БД.
2. Должно быть разработано несколько форм пользовательского интерфейса.
3. Разработчик самостоятельно определяет интерфейс программы и ее функциональность, однако для получения максимальной оценки приложение в обязательном порядке независимо от предметной области, указанной в задании, должно выполнять следующие операции:

* Отображать в сетках DataGridView данные предметной области:
* Для информационной модели, основанной на списках, на момент первого запуска программы допускается отсутствие файла. В этом случае списки объектов должны быть созданы в программе.
* Для информационной модели, основанной на БД, таблицы должны быть предварительно заполнены записями.
* Реализовать *добавление* в источник данных нового объекта, *удаление* объекта из источника данных, *редактирование* объекта в источнике данных.
* Реализовать фильтрацию записей источника данных, удовлетворяющих введенному пользователем сложному критерию.
* Реализовать сортировку записей источника данных.
* Для информационной модели, основанной на списках, сохранять источник данных в файле, используя XML-сериализацию**.**
* Для информационной модели, основанной на таблицах БД, обновлять изменения источника данных в базе данных.
* После сохранения данных при запуске программы загрузить данные из файла или базы данных.
* Используя меню или панель инструментов, вызвать приложение Блокнот для просмотра справки о программе: файл Help.txt текущего каталога программы.
* Создать пункт меню «Об авторе».
* Разработать несколько полезных пользователю функций для отображения статистических данных, например, средних, максимальных или минимальных значений, данных для построения гистограммы или графика и т.п.

1. Программа не должна завершаться аварийно: сообщения о некорректном вводе данных, противоречивых или недопустимых значениях данных, при отсутствии данных по функциональному запросу пользователя и других нештатных ситуациях отображать в окнах сообщений.
2. Программа должна быть читабельной и содержать полезные комментарии.
3. Список должен быть реализован в виде коллекции, например, динамического массива типа List<T> или BindingList<T>. Программа не должна завершаться аварийно: сообщения о некорректном вводе данных, противоречивых или недопустимых значениях данных, при отсутствии данных по функциональному запросу пользователя и других нештатных ситуациях отображать в окнах сообщений.
4. Программа должна быть читабельной и содержать полезные комментарии.

# 2. Описание предметной области

Предметной областью данного проекта является фирма, которая предоставляет автомобили в аренду. В работе фирмы присутствуют процессы, которые можно автоматизировать: бронирование автомобиля (изменение условий бронирования, отказ от бронирования), добавление нового автомобиля для аренды (добавляется администратором), изменение характеристик автомобиля и условий его аренды, удаление автомобиля, оставление клиентами отзывов и пожеланий, назначение нового администратора фирмы. Предоставление автомобилей в аренду является для заказчика основным видом деятельности, поэтому необходимо реализовать расчет и вывод статистических данных о доходах и деятельности фирмы.

В данном проекте автоматизированы все процессы, указанные выше.

# 3. Актуальность автоматизации

Клиенты фирмы являются основным источником дохода для заказчика, следовательно, их учет позволит оптимизировать затраты на некоторые виды предлагаемых в аренду автомобиле, а значит, увеличить прибыль фирмы. Также, важным для фирмы является мнение ее клиентов об арендованных автомобилях, поэтому возможность добавления пользователями приложения отзывов крайне актуальна.

Во время развития фирмы, она может приобретать новые автомобили и изменять цены на старые, следовательно, для того чтобы прайс лист приложения оставался актуальным, очень важно автоматизировать процесс добавления новых транспортных средств для аренды, а также их изменения и удаления.

Автоматизация указанных выше процессов деятельности фирмы позволит сократить непроизводительные методы использования труда работников, повысить производительность труда и, как следствие, увеличить объем предоставляемых услуг и качество обслуживания, что приведет к появлению новых клиентов и увеличению прибыли. Поэтому разработка приложения для автоматизации деятельности данной фирмы является актуальной.

# 4. Описание программы

## 4.1. Алгоритмические решения

Данные приложения сохраняются в виде списка соответствующего класса в нескольких файлах:

* Файл Cars.xml используется для хранения актуальных автомобилей для аренды;
* Файл Books.xml используется для хранения данных о бронировании автомобилей;
* Файл Users.xml используется для хранения данных о пользователях приложения;
* Файл Reports.xml используется для хранения данных об отзывах клиентов фирмы;

При входе в приложение пользователь сразу может выбрать автомобиль для бронирования, нажав на соответствующую сроку в таблице DataGridView. Для открытия формы бронирования необходимо авторизоваться или пройти регистрацию.

После авторизации откроется форма бронирования, при заполнении который от нажатия на кнопку забронировать бронь сохранится в файл Books.xml. Поля имени и email заполняются автоматически, в зависимости от текущего пользователя.

Если авторизованный пользователь является администратором, то при нажатии на строку в таблице автомобилей на начальном экране, откроется форма изменения и удаления выбранного автомобиля.

При внесении изменений в характеристики автомобиля или в условия его аренды появится кнопка сохранить изменения, при нажатии которых данные об автомобиле будут изменены в файле Cars.xml. При нажатии кнопки «Удалить» автомобиль будет удален из списка автомобилей в файле Cars.xml.

Просмотреть актуальные брони можно в личном кабинете, для входа в который необходимо авторизоваться (или зарегистрироваться).

В личном кабинете можно изменить или удалить актуальные брони, если до их начала осталось не менее 5-ти дней, в противном случае бронь можно только удалить. Изменению доступны только даты аренды.

Если пользователь является администратором, то ему доступны бронирования всех пользователей, и он может их изменить, если бронь еще не началась. Администратор может назначать других пользователей администраторами, для этого в личном кабинете есть таблица пользователей с возможностью изменения. Также, администратор в личном кабинете может добавлять автомобили и просматривать статистику.

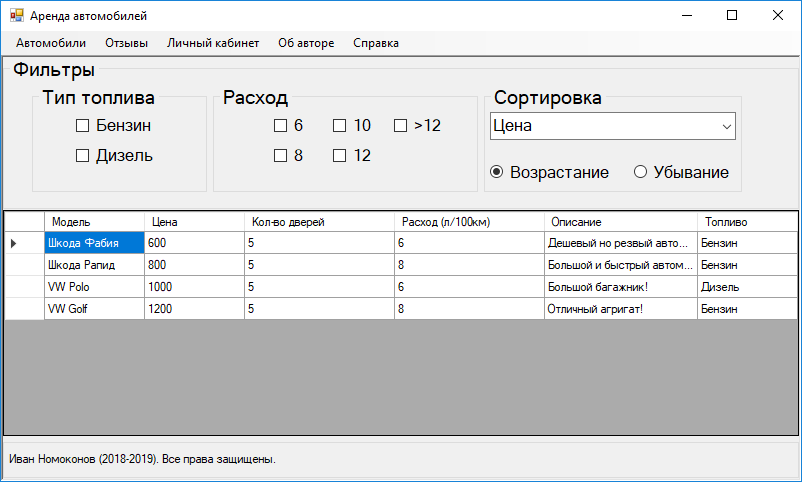
В разделе отзывов все пользователи могут добавить отзыв, предварительно авторизовавшись. На форме добавления отзыва необходимо выбрать автомобиль, к которому относится отзыв, а также указать оценку. Поля имени и email заполняются автоматически, в зависимости от текущего пользователя.

## 4.2. Описание интерфейса программы

В приложении для реализации интерфейса используется 13 элементов управления и компонентов:

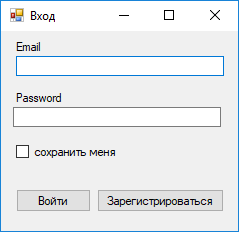
* DataGridView;
* Button;
* TextBox;
* Label;
* CheckBox
* RadioButton;
* GroupBox;
* DateTimePicker;
* MenuStrip;
* NumericUpDown;
* ComboBox;
* Panel.

На основных страницах приложения присутствует меню (MenuStrip) для перехода между основными страницами приложения.



На начальной странице (рис. 1) приложения присутствуют четыре GroupBox для фильтров и сортировки автомобилей, а также таблица dataGridView для вывода автомоблей в зависимости от фильтров и сортировки.

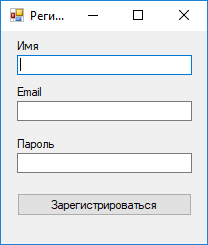
Для выбора фильтра необходимо отметить соответствующий CheckBox. Для активации сортировки необходима с помощью ComboBox выбрать поле, по которому нужно выполнить сортировку, а затем выбрать направление сортировки с помощью RadioButton.



При нажатии на ячейку таблицы автомобилей или при нажатии на пункт меню «Личный кабинет» откроется форма для входа пользователя. Авторизация необходима для оформления заказа.

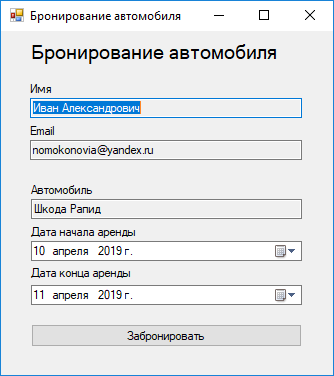
Если отметить CheckBox «сохранить меня», то пользователь сохранится после входа и при следующем открытии приложения будет авторизован автоматически.

Для регистрации нового пользователя необходимо на жать на кнопку «Зарегистрироваться». Откроется форма регистрации, где необходимо заполнить все поля для успешной регистрации.



После нажатия кнопки «Зарегистрироваться» будет создан новый пользователь и форма регистрации закроется. Вы сразу сможете войти в новый аккаунт.

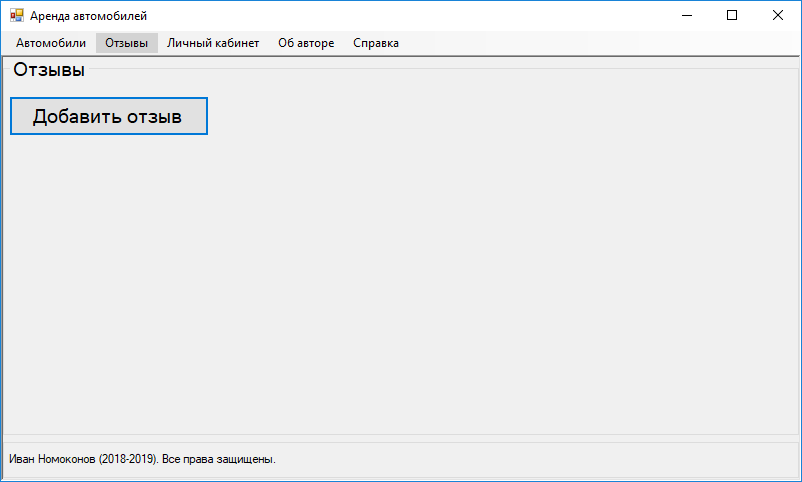
Если пользователь авторизован, то при нажатии на ячейку в таблице автомобилей откроется форма бронирования автомобиля (если пользователь не является администратором).



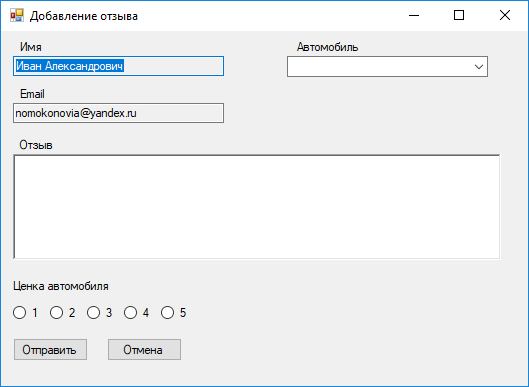
Имя, Email заказчика, а также модель автомобиля будут заполнены автоматически. На форме можно выбрать даты начала и конца аренды. Если дата конца будет меньше даты начала, то высветится соответствующая подсказка. Если в указанные даты автомобиль занят, будет высветится советующая подсказка. Подсказки высвечиваются после нажатия кнопки забронировать и убираются в случае любого изменения любой даты.

После успешного бронирования, новый заказ можно просмотреть, изменить и удалить в личном кабинете.

При нажатии на вкладку меню «Отзывы» откроется страница отзывов.



При нажатии на кнопку добавить отзыв, если пользователь авторизован откроется форма добавления отзыва, иначе откроется форма авторизации.

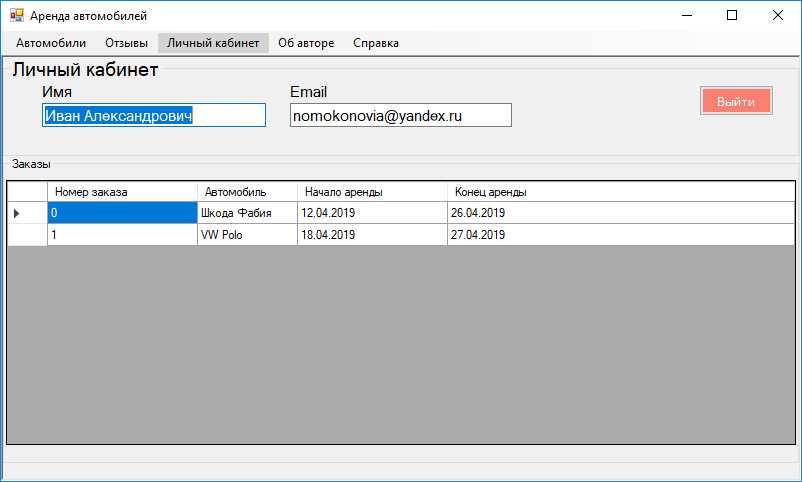


На форме добавления отзыва необходимо заполнить поле «Отзыв» и выбрать автомобиль к которому относится отзыв из выпадающего списка.

Поля Имя и Email заполняются автоматически в зависимости от текущего пользователя.

Также, обязательно необходимо поставить оценку автомобилю, иначе при попытке отправить отзыв выскочит подсказка «Заполните все поля».

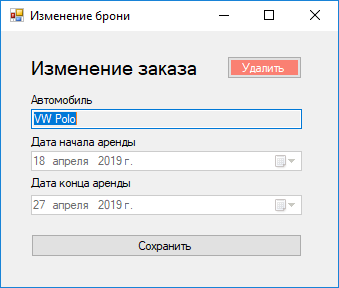
Войти в личный кабинет можно нажав на вкладку «Личный кабинет» в главном меню программы. Вход будет произведен, если пользователь авторизован, иначе откроется форма входа.



В личном кабинете обычный пользователь может просматривать, изменять и удалять свои заказы.

Чтобы изменить заказ необходимо нажать на любую ячейку в строке необходимого заказ.

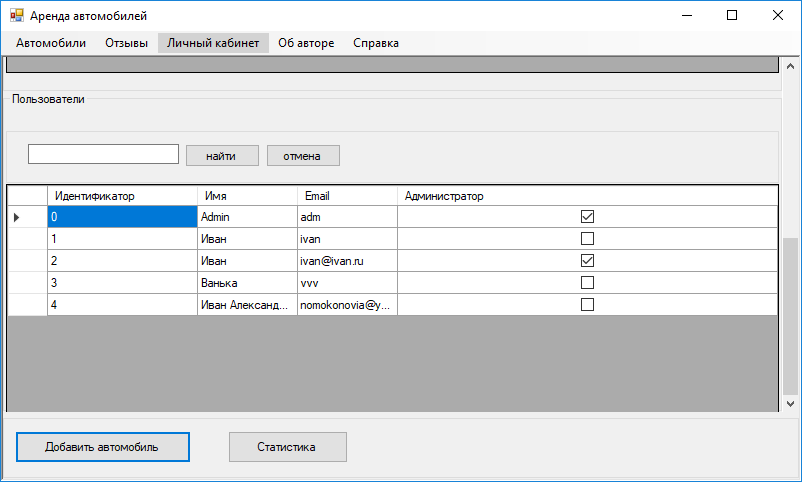
На форме изменения заказ можно изменить даты заказа или удалить заказ. Изменение дат возможно не позднее чем за 5 дней до даты начала заказа. Удаление не позднее одного дня до даты начала заказа.



Также в личном кабинете можно выйти из профиля (очистить текущего пользователя), нажав на кнопку «Выйти».

В личном кабинете администратора выводятся все заказы отсортированные по дате аренды.

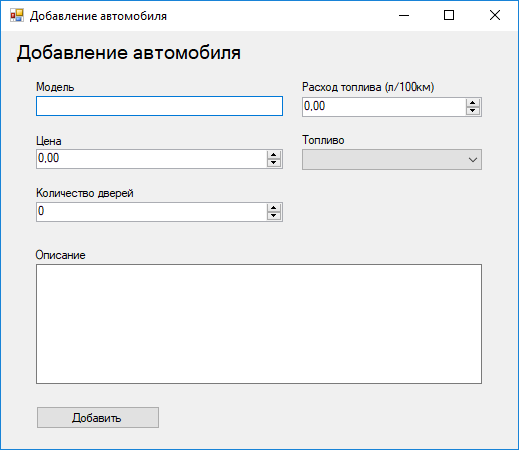
Так же администратор может назначить других пользователей администраторами. Для этого у него в личном кабинете есть таблица пользователей с возможностью поиска по email. Чтобы назначить пользователя администратором необходимо поставить галочку в соответствующей строке.



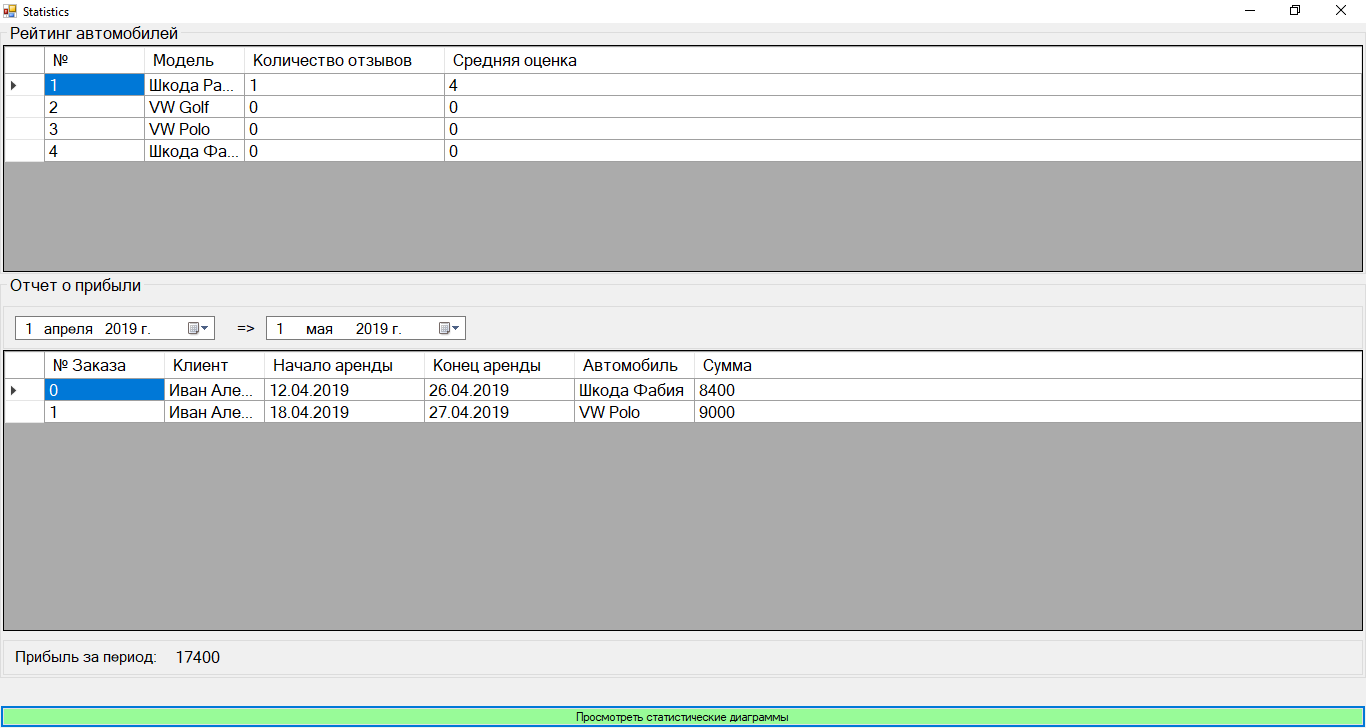
Администратор может изменять данные в таблице (кроме идентификатора).

Для поиска пользователь необходимо ввести его email в строку поиска и нажать на кнопку «найти», для показа всех пользователей нужно нажать «отмена».

У администратора в личном кабинете есть возможность добавить автомобиль. Для этого необходимо нажать на кнопку «Добавить автомобиль». После нажатия откроется форма добавления автомобиля, которой, для успешного добавления, необходимо заполнить все поля.

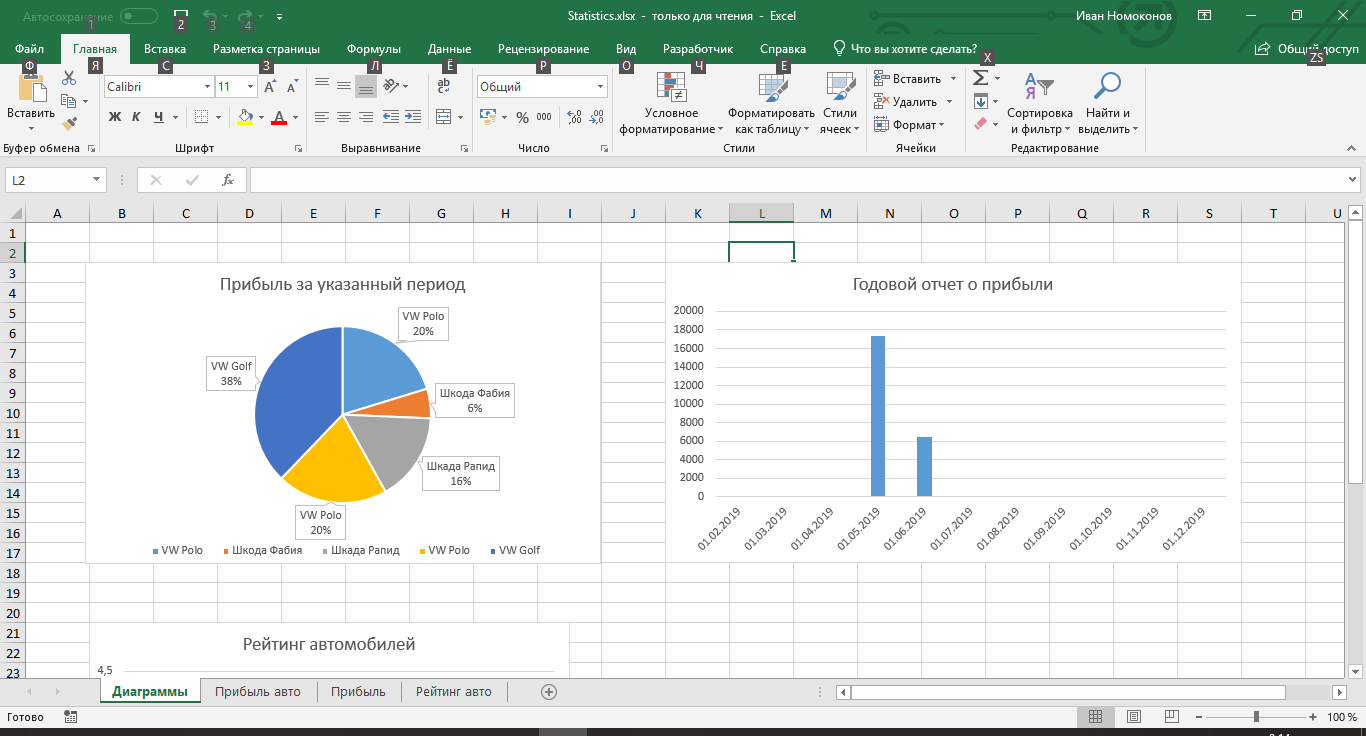


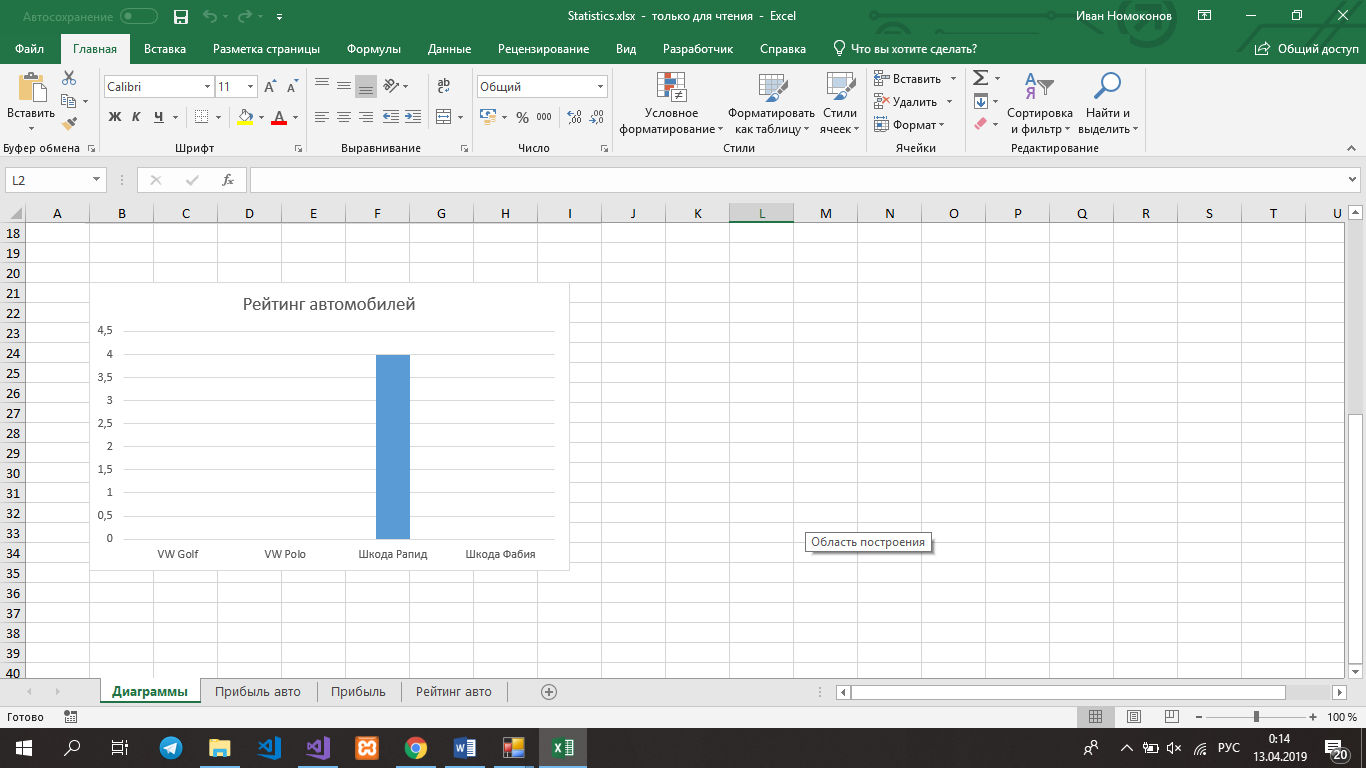
Также администратор может просмотреть статистику продаж, нажав на кнопку статистика в личном кабинете. Откроется окно статистика, где есть статистические данные по общему рейтингу автомобилей с указанием количества отзывов и средней оценки.



А также сводная таблица о заказах за определенный период с указанием полной суммы каждого заказа. После таблицы о прибыли указана общая сумма прибыли за период.

В низу формы статистики, есть кнопка «Просмотреть статистические диаграммы». По нажатию на эту кнопку будет открыт файл Excel с выводом графиков: годовой отчет о прибыли и рейтинг автомобилей.





## 4.3. Состав приложения

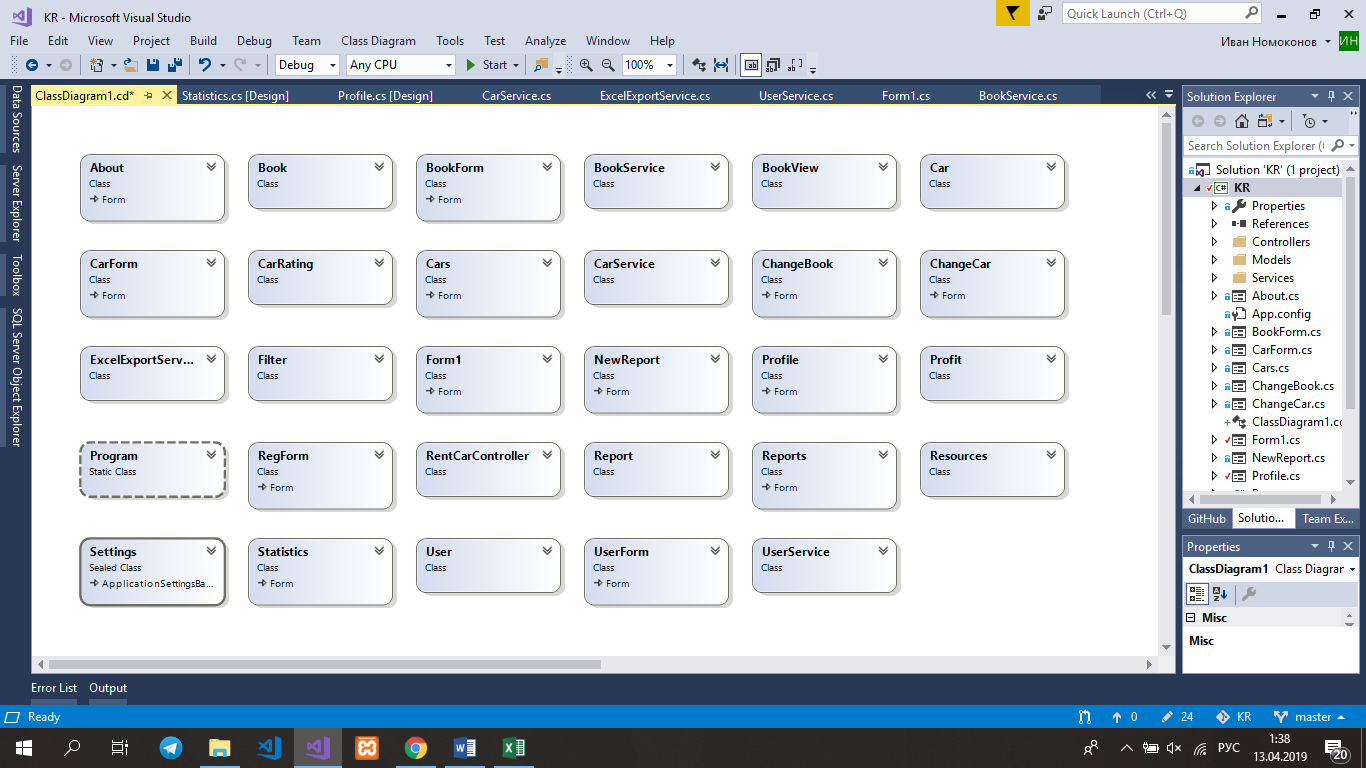
В состав приложения входят следующие файлы и папки:

* KR.exe – программа;
* Statistics.xlsx – статистические диаграммы
* Readme.txt–инструкция:
* Storage – папка с xml файлами для хранения данных приложения:
* Users.xml – все пользователи
* Cars.xml - автомобили
* Books.xml - бронирования
* Reports.xml - отзывы
* User.xml – текущий пользователь

# 5. Назначение и состав классов программы

## 5.1. Диаграмма классов

Разрабатываемая программа состоит из нескольких классов, содержащих множество полей, методов и свойств.



## 5.2. Классы приложения

Определение класса **User**клиента фирмы представлено в Приложении. В классе описаны следующие открытые свойства и конструктор:

* UserId – свойство, позволяющее идентифицировать каждого клиента;
* Name – свойство, позволяющее возвращать и задавать имя клиента;
* Surname – свойство, позволяющее возвращать и задавать фамилию клиента;
* Name – свойство, позволяющее возвращать и задавать имя клиента;
* Email – свойство, позволяющее возвращать и задавать электронную почту клиента;
* User() – конструктор по умолчанию;

Определение класса **Report**отзыва об автомобиле представлено в Приложении. В классе описаны следующие открытые свойства и конструктор:

* ReportId – свойство, позволяющее идентифицировать каждый отзыв;
* Car – свойство, позволяющее возвращать и задавать автомобиль на который сделать отзыв;
* User – свойство, позволяющее возвращать и задавать клиента, который оставил отзыв;
* Text – свойство, позволяющее возвращать и задавать текст отзыва;
* Mark – свойство, позволяющее возвращать и задавать электронную оценку автомобиля;
* CreateDate – свойство, позволяющее задавать и возвращать дату создания отзыва.
* Report() – конструктор по умолчанию;
* Report(параметры) – перегруженный контруктор с параметрами

Определение класса **Book** бронирования автомобиляпредставлено в Приложении. В классе описаны следующие открытые свойства и конструктор:

* BookId – свойство, позволяющее идентифицировать каждого клиента;
* User – свойство, позволяющее возвращать и задавать клиента осуществившего бронирование;
* Car – свойство, позволяющее возвращать и задавать автомобиль, который забронировали;
* DateStart – свойство, позволяющее возвращать и задавать дату начала аренды;
* DateFinish – свойство, позволяющее возвращать и задавать дату конца аренды;
* Book() – конструктор по умолчанию;

Определение класса **Car**автомобиля фирмы представлено в Приложении. В классе описаны следующие открытые свойства и конструктор:

* CarId – свойство, позволяющее идентифицировать каждый автомобиль;
* Model – свойство, позволяющее возвращать и задавать модель автомобиля;
* Price – свойство, позволяющее возвращать и задавать цену аренды автомобиля;
* Doors – свойство, позволяющее возвращать и задавать количество дверей автомобиля;
* Consumption – свойство, позволяющее возвращать и задавать расход топлива на 100км;
* Fuel – свойство позволяющее задавать и возвращать тип топлива автомобиля
* Description – свойство, позволяющее задавать и возвращать описание автомобиля
* Car() – конструктор по умолчанию;
* Car(параметры) – конструктор с параметрами

## 5.3. Служебные классы

К служебным классам относятся:

* **UserService –** класс для работы с файлом Users.xml. В данном классе реализованы методы для добавления и изменения пользователя, метод для получения всех пользователей, метод для получения одного пользователя по email и паролю, метод для сохранения текущего пользователя, а также методы для извлечения пользователей из xml файла и сохранения списка пользователей в xml файле. Здесь же реализован метод для получения уникального идентификатора для добавления нового пользователя.
* **CarService –** класс для работы с файлом Cars.xml. В данном классе реализованы методы для сериализации и десериализации списка автомобилей в файл Cars.xml, методы для добавления, изменения и удаления автомобилей. Также здесь реализованы методы для получения и добавления отзывов. Здесь же реализован метод для получения уникального идентификатора для добавления нового автомобиля.
* **BookService –** класс для работы с файлом Books.xml. Здесь реализованы методы: для получения всех броней, для получения броней конкретного пользователя, для изменения брони, для удаления брони, для добавления новой брони. Здесь же реализован метод для получения уникального идентификатора для добавления новой брони.
* **ExcelExportService –** класс для вывода статистических данных в файл Statistics.xlsx для построения в нем диаграмм. В данном классе реализованы методы для заполнения таблиц эксель средними оценками автомобилей фирмы на основе отзывов о них, а также прибылью в указанный промежуток времени.
* **RentCarController –** класс, который создан, чтобы объединить сервисы по взаимодействую с xml и xlsx файла, что улучшить логику приложения. Здесь реальзованы перегруженные методы для вызова соответствующих методов сервисов.
* **Program** – класс, содержащий метод Main, который позволяет запустить программу, открывая главную форму Form1.

## 5.4. Формы

Ниже представлены пользовательские классы, наследующие базовую функциональность от класса Form.

* **Form1**– класс, который является mdi контейнером и содержит основное меню программы и форму в соответствии с выбранным пунктом меню.
* **Cars –** класс представляющий стартовую страницу приложения, где выведены автомобили, фильтры и сортировка
* **Reports –** класс представляющий страницу отзывов
* **Profile –** класс представляющий личный кабинет
* **About –** класс представляющий страницу об авторе
* **UserForm –** класс представляющий форму входа
* **RegForm –** класс представляющий форму регистрации
* **CarForm –** класс, представляющий форму добавления автомобиля
* **BookForm –** класс, представляющий форму для аренды автомобиля
* **NewReport –** класс, представляющий форму для добавления отзыва
* **ChangeBook -**  класс представляющий форму для изменения бронирования
* **ChangeCar** – класс, представляющий форму для изменения автомобиля
* **Statistics** – класс, представляющий форму для вывода статистической информации.

# Заключение

Разработав Windows приложение, был выполнен ряд поставленных задач. Справочно-информационная система написана на языке программирования С# с использованием классов, методов и свойств. Для реализации некоторых задач потребовалось обратиться к объектам операционной системы: файловая система, процессы.

Программа может модернизироваться и обновляться. К примеру, может быть расширен класс клиента путем добавления паспортных данных, денежных сумм предоплаты и итоговой платы, примечания и т.д.

Программа может быть расширена статистическими и финансовыми функциями.

# Список литературы

1. Горелов С.В., Волков А.Г. Разработка Windows-приложений. Часть 1. Учебное пособие. Образовательный портал Финансового университета. 2018.
2. Горелов С.В. Разработка Windows-приложений. Часть 2. Учебное пособие. Образовательный портал Финансового университета. 2018.
3. Официальный сайт Microsoft: [Интернет-ресурс]. URL: https://msdn.microsoft.com org.

# Приложение. Исходный код программы

**Папка Models (основные модели)**

**User.cs**

namespace KR.Models

{

public class User

{

public int UserId { get; set; }

public string Name { get; set; }

public string Email { get; set; }

public string Password { get; set; }

public bool IsAdmin { get; set; }

}

}

**Book.cs**

namespace KR.Models

{

public class Book

{

public int BookId { get; set; }

public Car Car { get; set; }

public User User { get; set; }

public DateTime DateStart { get; set; }

public DateTime DateFinish { get; set; }

}

}

**Car.cs**

namespace KR.Models

{

public class Car

{

public int CarId { get; set; }

public string Model { get; set; }

public decimal Price { get; set; }

public int Doors { get; set; }

public decimal Consumption { get; set; }

public string Description { get; set; }

public string Fuel { get; set; }

}

}

**Report.cs**

namespace KR.Models

{

public class Report

{

public int ReportId { get; set; }

public Car Car { get; set; }

public User User { get; set; }

public string Text { get; set; }

public int Mark { get; set; }

public DateTime CreateDate { get; set; }

}

}

**Папка Services**

**UserService.cs**

using KR.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.IO;

using System.Xml.Serialization;

namespace KR.Services

{

class UserService

{

public List<User> Users { get; set; }

public UserService()

{

Users = GetUsers();

}

public List<User> GetUsers()

{

if (File.Exists("Storage/Users.xml"))

{

// Десериализуем

XmlSerializer xs = new XmlSerializer(typeof(List<User>));

FileStream fs = new FileStream("Storage/Users.xml", FileMode.Open);

var Users = (List<User>)xs.Deserialize(fs);

fs.Close();

return Users;

}

else

{

return new List<User>();

}

}

public User GetUser(string email, string password, bool save = false)

{

var p = Users.Find(u => u.Email == email && u.Password == password);

if (save)

{

SaveUser(p);

}

return p;

}

public User GetUser()

{

if (File.Exists("Storage/User.xml"))

{

// Десериализуем

XmlSerializer xs = new XmlSerializer(typeof(User));

FileStream fs = new FileStream("Storage/User.xml", FileMode.Open);

var User = (User)xs.Deserialize(fs);

fs.Close();

return User;

}

else

{

return null;

}

}

public void AddUser(User user)

{

Users.Add(user);

Save();

}

public void Update(User user)

{

var u = Users.Find(x => x.UserId == user.UserId);

u.Name = user.Name;

u.Email = user.Email;

u.IsAdmin = user.IsAdmin;

Save();

}

public void SaveUser(User user)

{

if (File.Exists("Storage/User.xml"))

{

File.WriteAllText("Storage/User.xml", "");

}

XmlSerializer xs = new XmlSerializer(typeof(User));

FileStream fs = new FileStream("Storage/User.xml", FileMode.OpenOrCreate);

xs.Serialize(fs, user);

fs.Close();

}

public void Exit()

{

if (File.Exists("Storage/User.xml"))

{

File.Delete("Storage/User.xml");

}

}

public void Save()

{

if (File.Exists("Storage/Users.xml"))

{

File.WriteAllText("Storage/Users.xml", "");

}

XmlSerializer xs = new XmlSerializer(typeof(List<User>));

FileStream fs = new FileStream("Storage/Users.xml", FileMode.OpenOrCreate);

xs.Serialize(fs, Users);

fs.Close();

}

public int GetNewUserId()

{

return Users.Count;

}

}

}

**CarService.cs**

using KR.Models;

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Xml.Serialization;

namespace KR.Services

{

class CarService

{

public List<Report> Reports { get; set; }

public List<Car> Cars { get; set; }

public CarService()

{

Reports = GetReports();

Cars = GetCars();

}

public List<Report> GetReports()

{

if (File.Exists("Storage/Reports.xml"))

{

// Десериализуем

XmlSerializer xs = new XmlSerializer(typeof(List<Report>));

FileStream fs = new FileStream("Storage/Reports.xml", FileMode.Open);

var Reports = (List<Report>)xs.Deserialize(fs);

fs.Close();

return Reports;

}

else

{

return new List<Report>();

}

}

public List<Report> GetCarReports(int CarId)

{

return Reports.FindAll(r => r.Car.CarId == CarId);

}

public List<int> GetCarMarks(int CarId)

{

return Reports.FindAll(r => r.Car.CarId == CarId).Select(r => r.Mark).ToList();

}

public List<Car> GetCars()

{

if (File.Exists("Storage/Cars.xml"))

{

// Десериализуем

XmlSerializer xs = new XmlSerializer(typeof(List<Car>));

FileStream fs = new FileStream("Storage/Cars.xml", FileMode.Open);

var Cars = (List<Car>)xs.Deserialize(fs);

fs.Close();

return Cars;

}

else

{

return new List<Car>();

}

}

public void AddReport(Report report)

{

Reports.Add(report);

SaveReports();

}

public void Update(Car car)

{

var c = Cars.Find(x => x.CarId == car.CarId);

c.Model = car.Model;

c.Price = car.Price;

c.Doors = car.Doors;

c.Consumption = car.Consumption;

c.Fuel = car.Fuel;

c.Description = car.Description;

Save();

}

public int GetNewReportId()

{

return Reports.Count;

}

public void AddCar(Car car)

{

Cars.Add(car);

Save();

}

private void Save()

{

if (File.Exists("Storage/Cars.xml"))

{

File.WriteAllText("Storage/Cars.xml", "");

}

XmlSerializer xs = new XmlSerializer(typeof(List<Car>));

FileStream fs = new FileStream("Storage/Cars.xml", FileMode.OpenOrCreate);

xs.Serialize(fs, Cars);

fs.Close();

}

private void SaveReports()

{

if (File.Exists("Storage/Reports.xml"))

{

File.WriteAllText("Storage/Reports.xml", "");

}

XmlSerializer xs = new XmlSerializer(typeof(List<Report>));

FileStream fs = new FileStream("Storage/Reports.xml", FileMode.OpenOrCreate);

xs.Serialize(fs, Reports);

fs.Close();

}

public void Remove(Car car)

{

var i = Cars.Select(x => x.CarId).ToList().IndexOf(car.CarId);

Cars.RemoveAt(i);

Save();

}

public int GetNewCarId()

{

return Cars.Count;

}

}

}

**BookService.cs**

using KR.Models;

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Xml.Serialization;

namespace KR.Services

{

class BookService

{

public List<Book> Books;

public BookService()

{

Books = Get();

}

public List<Book> Get()

{

if (File.Exists("Storage/Books.xml"))

{

// Десериализуем

XmlSerializer xs = new XmlSerializer(typeof(List<Book>));

FileStream fs = new FileStream("Storage/Books.xml", FileMode.Open);

var Books = (List<Book>)xs.Deserialize(fs);

fs.Close();

return Books;

}

else

{

return new List<Book>();

}

}

public List<Book> GetCarBooks(int carId)

{

return Books.FindAll(b => b.Car.CarId == carId);

}

public List<Book> GetUserBooks(int userId)

{

return Books.FindAll(b => b.User.UserId == userId);

}

public void Add(Book book)

{

Books.Add(book);

Save();

}

public bool CheckCarBook(DateTime dateStart, DateTime dateFinish, int carId)

{

var books = GetCarBooks(carId);

foreach(Book b in books)

{

if(b.DateStart<dateStart && b.DateFinish > dateStart)

{

return false;

}

if (b.DateStart < dateFinish && b.DateFinish > dateStart)

{

return false;

}

}

return true;

}

public void Update(Book book)

{

Book b = Books.Find(x => x.BookId == book.BookId);

b.DateStart = book.DateStart;

b.DateFinish = book.DateFinish;

Save();

}

public void Remove(Book book)

{

Books.Remove(book);

Save();

}

private void Save()

{

if (File.Exists("Storage/Books.xml"))

{

File.WriteAllText("Storage/Books.xml", "");

}

XmlSerializer xs = new XmlSerializer(typeof(List<Book>));

FileStream fs = new FileStream("Storage/Books.xml", FileMode.OpenOrCreate);

xs.Serialize(fs, Books);

fs.Close();

}

public int GetNewBookId()

{

return Books.Count;

}

}

}

**ExcelExportSevice.cs**

using KR.Controllers;

using KR.Models;

using Microsoft.Office.Interop.Excel;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace KR.Services

{

class ExcelExportService

{

private Application excelapp;

private Worksheet excelworksheet;

private RentCarController \_ctrl;

private string fileName = System.Windows.Forms.Application.StartupPath + "\\Statistics.xlsx";

public ExcelExportService()

{

Open();

}

private void Open()

{

excelapp = new Application();

excelapp.Visible = false;

excelapp.Workbooks.Open(fileName);

}

public void Show(List<CarRating> rc, RentCarController ctrl)

{

\_ctrl = ctrl;

ShowProfit();

ShowRating();

excelapp.Visible = true;

}

private void ShowRating()

{

excelworksheet = (Worksheet)excelapp.Worksheets.get\_Item(4);

var cars = \_ctrl.GetCars();

for (int m = 0; m < cars.Count; m++)

{

var excelcells = (Range)excelworksheet.Cells[m+2, 1];

//выводим координаты ячеек

excelcells.Value2 = cars[m].Model;

excelcells = (Range)excelworksheet.Cells[m+2, 2];

var marks = \_ctrl.GetCarMarks(cars[m].CarId);

if (marks.Count > 0)

{

excelcells.Value2 = marks.Average();

}

else

{

excelcells.Value2 = 0;

}

}

}

private void ShowProfit()

{

excelworksheet = (Worksheet)excelapp.Worksheets.get\_Item(3);

var year = DateTime.Now.Year;

for (int m = 2; m < 13; m++)

{

var date1 = new DateTime(year, m-1, 1);

var date2 = new DateTime(year, m, 1);

var excelcells = (Range)excelworksheet.Cells[m, 1];

//выводим координаты ячеек

excelcells.Value2 = date2.ToShortDateString();

excelcells = (Range)excelworksheet.Cells[m, 2];

excelcells.Value2 = \_ctrl.GetB().Where(x => x.DateStart<date2 && x.DateStart>date1).Select(x => x.Car.Price\*(x.DateFinish-x.DateStart).Days).Sum();

}

}

}

}

**Папка Controllers**

**RentCarController.cs**

using KR.Models;

using KR.Services;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace KR.Controllers

{

public class RentCarController

{

private UserService us = new UserService();

private CarService cs = new CarService();

private BookService bs = new BookService();

private ExcelExportService export = new ExcelExportService();

public void Add(User user)

{

us.AddUser(user);

}

public User GetUser(string email, string password, bool save = false)

{

return us.GetUser(email, password, save);

}

public List<User> GetUsers()

{

return us.GetUsers();

}

public User GetU()

{

return us.GetUser();

}

public List<Book> GetBooks()

{

return bs.Get();

}

public List<Book> GetB()

{

return bs.Books;

}

public int GetNewUserId()

{

return us.GetNewUserId();

}

public void Exit()

{

us.Exit();

}

public int GetNewReportId()

{

return cs.GetNewReportId();

}

public void Add(Report r)

{

cs.AddReport(r);

}

public List<Report> GetReports()

{

return cs.GetReports();

}

public List<Report> GetCarReports(int CarId)

{

return cs.GetCarReports(CarId);

}

public List<int> GetCarMarks(int CarId)

{

return cs.GetCarMarks(CarId);

}

public void Add(Car c)

{

cs.AddCar(c);

}

public int GetNewCarId()

{

return cs.GetNewCarId();

}

public List<Car> GetCars()

{

return cs.GetCars();

}

public int GetNewBookId()

{

return bs.GetNewBookId();

}

public void Add(Book book)

{

bs.Add(book);

}

public void Update(Book book)

{

bs.Update(book);

}

public void Update(User user)

{

us.Update(user);

}

public bool CheckCarBook(DateTime dateStart, DateTime dateFinish, int carId)

{

return bs.CheckCarBook(dateStart, dateFinish, carId);

}

public void Update(Car car)

{

cs.Update(car);

}

public void Remove(Book book)

{

bs.Remove(book);

}

public void Remove(Car car)

{

cs.Remove(car);

}

public List<Book> GetUserBooks(int UserId)

{

return bs.GetUserBooks(UserId);

}

public void Export(List<CarRating> rc)

{

export.Show(rc, this);

}

}

}

**Формы**

**About.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace KR

{

public partial class About : Form

{

public About()

{

InitializeComponent();

textBox1.Focus();

}

}

}

**BookForm.cs**

using KR.Models;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace KR

{

public partial class BookForm : Form

{

private Cars parent;

private Car car;

public BookForm(Cars parent, Car car)

{

this.parent = parent;

this.car = car;

InitializeComponent();

dateTimePicker1.Value = new DateTime(DateTime.Today.Ticks + TimeSpan.TicksPerDay);

dateTimePicker2.Value = new DateTime(DateTime.Today.Ticks + TimeSpan.TicksPerDay\*2);

textBox4.Text = this.car.Model;

textBox1.Text = this.parent.parent.user.Name;

textBox2.Text = this.parent.parent.user.Email;

}

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

{

if (dateTimePicker1.Value < DateTime.Now || dateTimePicker2.Value <= dateTimePicker1.Value)

{

note.Visible = true;

note2.Visible = false;

return;

}

if(!parent.ctrl.CheckCarBook(dateTimePicker1.Value.Date, dateTimePicker2.Value.Date, car.CarId))

{

note2.Visible = true;

note.Visible = false;

return;

}

Book book = new Book

{

BookId = parent.ctrl.GetNewBookId(),

User = parent.parent.user,

Car = car,

DateStart = dateTimePicker1.Value.Date,

DateFinish = dateTimePicker2.Value.Date

};

parent.AddBook(book);

}

private void clearNote(object sender = null, EventArgs e = null)

{

note.Visible = false;

}

}

}

**CarForm.cs**

using KR.Controllers;

using KR.Models;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace KR

{

public partial class CarForm : Form

{

public Profile parent;

public RentCarController ctrl;

public CarForm()

{

InitializeComponent();

}

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

{

if(model.Text == "" ||

price.Value == 0 ||

doors.Value==0 ||

description.Text == "" ||

consumption.Value == 0 ||

fuel.Text == "")

{

note.Visible = true;

return;

}

Car car = new Car

{

CarId = ctrl.GetNewCarId(),

Model = model.Text,

Price = price.Value,

Doors = (int)doors.Value,

Description = description.Text,

Consumption = consumption.Value,

Fuel = fuel.Text

};

parent.AddCar(car);

}

private void clearNote(object sender = null, EventArgs e = null)

{

note.Visible = false;

}

}

}

**Cars.cs**

using KR.Controllers;

using KR.Models;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace KR

{

public partial class Cars : Form

{

public RentCarController ctrl;

public Form1 parent;

private List<Car> cars;

private List<Car> curCars;

private BookForm bookForm;

private ChangeCar chCarForm;

private List<Filter> filters = new List<Filter>();

private string SortType = "Цена";

private bool SortDir = true;

public Cars(RentCarController ctrl, Form1 parent)

{

this.ctrl = ctrl;

this.parent = parent;

InitializeComponent();

label1.TextAlign = ContentAlignment.MiddleCenter;

cars = ctrl.GetCars();

comboBox1.Text = "Цена";

radioButton1.Checked = true;

}

private void changeSortType(object sender, EventArgs e)

{

SortType = comboBox1.Text;

Sort();

}

private void checkBox2\_CheckedChanged(object sender, EventArgs e)

{

var f = filters.Find(f1 => f1.Name == "Fuel" && f1.Value == "Дизель");

if (null == f)

{

filters.Add(new Filter { Name = "Fuel", Value = "Дизель" });

}

else

{

filters.Remove(f);

}

Filter();

}

private void FillCars(List<Car> cars)

{

curCars = cars;

carBindingSource.DataSource = cars;

}

private void dataGridView1\_CellMouseClick(object sender, DataGridViewCellMouseEventArgs e)

{

if (e.RowIndex<0)

{

return;

}

if(parent.user == null)

{

parent.showForm();

}

else

{

if (parent.user.IsAdmin)

{

chCarForm = new ChangeCar(this, curCars[e.RowIndex]);

chCarForm.Show();

}

else

{

bookForm = new BookForm(this, curCars[e.RowIndex]);

bookForm.Show();

}

}

}

public void CloseForm()

{

bookForm.Hide();

}

public void CloseCarForm()

{

FillCars(cars = ctrl.GetCars());

chCarForm.Hide();

}

public void AddBook(Book book)

{

ctrl.Add(book);

CloseForm();

}

private void checkBox7\_CheckedChanged(object sender, EventArgs e)

{

var f = filters.Find(f1 => f1.Name == "Consumption" && f1.Value == "100");

if (null == f)

{

filters.Add(new Filter { Name = "Consumption", Value = "100" });

}

else

{

filters.Remove(f);

}

Filter();

}

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

{

var f = filters.Find(f1 => f1.Name == "Fuel" && f1.Value == "Бензин");

if (null == f)

{

filters.Add(new Filter { Name = "Fuel", Value = "Бензин" });

}

else

{

filters.Remove(f);

}

Filter();

}

private void cons6\_CheckedChanged(object sender, EventArgs e)

{

var f = filters.Find(f1 => f1.Name == "Consumption" && f1.Value == "6");

if (null == f)

{

filters.Add(new Filter { Name = "Consumption", Value = "6" });

}

else

{

filters.Remove(f);

}

Filter();

}

private void cons8\_CheckedChanged(object sender, EventArgs e)

{

var f = filters.Find(f1 => f1.Name == "Consumption" && f1.Value == "8");

if (null == f)

{

filters.Add(new Filter { Name = "Consumption", Value = "8" });

}

else

{

filters.Remove(f);

}

Filter();

}

private void cons10\_CheckedChanged(object sender, EventArgs e)

{

var f = filters.Find(f1 => f1.Name == "Consumption" && f1.Value == "10");

if (null == f)

{

filters.Add(new Filter { Name = "Consumption", Value = "10" });

}

else

{

filters.Remove(f);

}

Filter();

}

private void cons12\_CheckedChanged(object sender, EventArgs e)

{

var f = filters.Find(f1 => f1.Name == "Consumption" && f1.Value == "12");

if (null == f)

{

filters.Add(new Filter { Name = "Consumption", Value = "12" });

}

else

{

filters.Remove(f);

}

Filter();

}

private void Filter()

{

if (filters.Count == 0)

{

FillCars(cars);

}

else

{

curCars = new List<Car>();

foreach (Car c in cars)

{

foreach (Filter f in filters)

{

if (c.Fuel == f.Value)

{

curCars.Add(c);

break;

}

if (f.Name == "Consumption" && c.Consumption == int.Parse(f.Value))

{

curCars.Add(c);

break;

}

}

}

FillCars(curCars);

}

}

private void radioButton1\_CheckedChanged(object sender, EventArgs e)

{

SortDir = true;

Sort();

}

private void radioButton2\_CheckedChanged(object sender, EventArgs e)

{

SortDir = false;

Sort();

}

private void Sort()

{

switch (SortType)

{

case "Цена":

{

if (SortDir)

{

FillCars(cars.OrderBy(x => x.Price).ToList());

}

else

{

FillCars(cars.OrderByDescending(x => x.Price).ToList());

}

break;

}

case "Модель":

{

if (SortDir)

{

FillCars(cars.OrderBy(x => x.Model).ToList());

}

else

{

FillCars(cars.OrderByDescending(x => x.Model).ToList());

}

break;

}

case "Расход":

{

if (SortDir)

{

FillCars(cars.OrderBy(x => x.Consumption).ToList());

}

else

{

FillCars(cars.OrderByDescending(x => x.Consumption).ToList());

}

break;

}

case "Количество дверей":

{

if (SortDir)

{

FillCars(cars.OrderBy(x => x.Doors).ToList());

}

else

{

FillCars(cars.OrderByDescending(x => x.Doors).ToList());

}

break;

}

}

}

}

}

**ChangeBook.cs**

using KR.Controllers;

using KR.Models;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace KR

{

public partial class ChangeBook : Form

{

public Book \_book { get; set; }

public Profile parent { get; set; }

public ChangeBook(Book book, Profile profile)

{

\_book = book;

parent = profile;

InitializeComponent();

textBox4.Text = book.Car.Model;

dateTimePicker1.Value = book.DateStart;

dateTimePicker2.Value = book.DateFinish;

if (book.DateStart < DateTime.Now || (book.DateStart - DateTime.Now).Days < 7)

{

dateTimePicker1.Enabled = false;

dateTimePicker2.Enabled = false;

}

}

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

{

if(dateTimePicker1.Value < DateTime.Now || dateTimePicker2.Value < dateTimePicker1.Value)

{

note.Visible = true;

return;

}

\_book.DateStart = dateTimePicker1.Value;

\_book.DateFinish = dateTimePicker2.Value;

parent.parent.ctrl.Update(\_book);

parent.CloseChangeForm();

}

private void clearNote(object sender, EventArgs e)

{

note.Visible = false;

}

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

{

parent.parent.ctrl.Remove(\_book);

parent.CloseChangeForm();

}

}

}

**ChangeCar.cs**

using KR.Models;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace KR

{

public partial class ChangeCar : Form

{

public Cars parent { get; set; }

public Car car { get; set; }

public ChangeCar(Cars parent, Car car)

{

this.parent = parent;

this.car = car;

InitializeComponent();

model.Text = car.Model;

price.Value = car.Price;

doors.Value = car.Doors;

consumption.Value = car.Consumption;

fuel.Text = car.Fuel;

description.Text = car.Description;

}

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

{

if (model.Text == "" ||

price.Value == 0 ||

doors.Value == 0 ||

description.Text == "" ||

consumption.Value == 0 ||

fuel.Text == "")

{

note.Visible = true;

return;

}

car.Model = model.Text;

car.Price = price.Value;

car.Doors = (int)doors.Value;

car.Consumption = consumption.Value;

car.Fuel = fuel.Text;

car.Description = description.Text;

parent.parent.ctrl.Update(car);

parent.CloseCarForm();

}

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

{

parent.parent.ctrl.Remove(car);

parent.CloseCarForm();

}

private void clearNote(object sender = null, EventArgs e = null)

{

note.Visible = false;

}

}

}

**Form1.cs**

using KR.Controllers;

using KR.Models;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Diagnostics;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace KR

{

public partial class Form1 : Form

{

public UserForm enterForm;

private Form curForm;

public User user = null;

public RentCarController ctrl = new RentCarController();

public Form1()

{

InitializeComponent();

this.IsMdiContainer = true;

Cars carsForm = new Cars(ctrl, this);

carsForm.FormBorderStyle = FormBorderStyle.None;

carsForm.MdiParent = this;

carsForm.WindowState = FormWindowState.Maximized;

carsForm.Dock = DockStyle.Fill;

this.curForm = carsForm;

curForm.Show();

user = ctrl.GetU();

автомобилиToolStripMenuItem.BackColor = Color.LightGray;

}

public void автомобилиToolStripMenuItem\_Click(object sender = null, EventArgs e = null)

{

if (curForm is Cars)

{

return;

}

curForm.Close();

Cars carsForm = new Cars(ctrl, this);

carsForm.MdiParent = this;

carsForm.FormBorderStyle = FormBorderStyle.None;

carsForm.Dock = DockStyle.Fill;

this.curForm = carsForm;

curForm.Show();

Clear();

автомобилиToolStripMenuItem.BackColor = Color.LightGray;

}

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

{

if(curForm is Reports)

{

return;

}

curForm.Close();

Reports reportsForm = new Reports(this);

reportsForm.MdiParent = this;

reportsForm.Dock = DockStyle.Fill;

reportsForm.FormBorderStyle = FormBorderStyle.None;

curForm = reportsForm;

curForm.Show();

Clear();

отзывыToolStripMenuItem.BackColor = Color.LightGray;

}

private void Clear()

{

личныйКабинетToolStripMenuItem.BackColor = SystemColors.Control;

автомобилиToolStripMenuItem.BackColor = SystemColors.Control;

отзывыToolStripMenuItem.BackColor = SystemColors.Control;

оСебеToolStripMenuItem.BackColor = SystemColors.Control;

}

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

{

if (curForm is Profile)

{

return;

}

if(user == null)

{

showForm();

}

else

{

curForm.Close();

Profile profileForm = new Profile(this);

profileForm.MdiParent = this;

profileForm.Dock = DockStyle.Fill;

profileForm.FormBorderStyle = FormBorderStyle.None;

curForm = profileForm;

curForm.Show();

Clear();

личныйКабинетToolStripMenuItem.BackColor = Color.LightGray;

}

}

public void showForm()

{

enterForm = new UserForm();

enterForm.parent = this;

enterForm.Show();

}

public void CloseForm()

{

enterForm.Hide();

}

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

{

if (curForm is About)

{

return;

}

curForm.Close();

About aboutForm = new About();

aboutForm.MdiParent = this;

aboutForm.MdiParent = this;

aboutForm.Dock = DockStyle.Fill;

aboutForm.FormBorderStyle = FormBorderStyle.None;

curForm = aboutForm;

curForm.Show();

Clear();

оСебеToolStripMenuItem.BackColor = Color.LightGray;

}

public void EnterUser(string email, string password, bool save)

{

user = ctrl.GetUser(email, password, save);

CloseForm();

}

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

{

Process p = Process.Start("notepad.exe", "readme.txt");

}

}

}

**NewReport.cs**

using KR.Controllers;

using KR.Models;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace KR

{

public partial class NewReport : Form

{

public Reports parent;

private List<Car> cars;

private int Mark = -1;

public NewReport(Reports parent)

{

InitializeComponent();

this.parent = parent;

textBox1.Text = parent.parent.user.Name;

textBox2.Text = parent.parent.user.Email;

cars = parent.parent.ctrl.GetCars();

foreach (Car c in cars)

{

comboBox1.Items.Add(c.Model);

}

}

private void label1\_Click(object sender, EventArgs e)

{

}

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

{

if (comboBox1.Text=="" || richTextBox1.Text == "" || Mark<0)

{

note.Visible = true;

return;

}

parent.AddReport(new Report

{

ReportId = parent.parent.ctrl.GetNewReportId(),

User = null,

Text = richTextBox1.Text,

Mark = Mark,

Car = cars.Find(x => x.Model==comboBox1.Text),

CreateDate = DateTime.Now

});

parent.CloseForm();

}

private void clearNote(object sender = null, EventArgs e = null)

{

note.Visible = false;

}

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

{

parent.CloseForm();

}

private void radioButton1\_CheckedChanged(object sender, EventArgs e)

{

Mark = 1;

clearNote();

}

private void radioButton2\_CheckedChanged(object sender, EventArgs e)

{

Mark = 2;

clearNote();

}

private void radioButton3\_CheckedChanged(object sender, EventArgs e)

{

Mark = 3;

clearNote();

}

private void radioButton4\_CheckedChanged(object sender, EventArgs e)

{

Mark = 4;

clearNote();

}

private void radioButton5\_CheckedChanged(object sender, EventArgs e)

{

Mark = 5;

clearNote();

}

}

}

**Profile.cs**

using KR.Controllers;

using KR.Models;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace KR

{

public partial class Profile : Form

{

public Form1 parent { get; set; }

private CarForm addCarForm;

private ChangeBook chBookForm;

private List<Book> books;

public Profile(Form1 p)

{

InitializeComponent();

parent = p;

name.Text = parent.user.Name;

email.Text = parent.user.Email;

if (!parent.user.IsAdmin)

{

groupBox2.Visible = false;

groupBox3.Visible = false;

}

ShowBooks();

}

private void ShowBooks()

{

bookViewBindingSource.Clear();

if (parent.user.IsAdmin)

{

books = parent.ctrl.GetBooks();

}

else

{

books = parent.ctrl.GetUserBooks(parent.user.UserId);

}

bookViewBindingSource.DataSource = books.Select(x => new BookView { BookId = x.BookId, Car = x.Car.Model, DateFinish = x.DateFinish, DateStart = x.DateStart });

}

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

{

addCarForm = new CarForm();

addCarForm.parent = this;

addCarForm.ctrl = parent.ctrl;

addCarForm.Show();

}

public void CloseForm()

{

addCarForm.Hide();

}

public void CloseChangeForm()

{

ShowBooks();

chBookForm.Hide();

}

public void AddCar(Car car)

{

parent.ctrl.Add(car);

CloseForm();

}

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

{

parent.user = null;

parent.ctrl.Exit();

parent.автомобилиToolStripMenuItem\_Click();

}

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

{

var st = new Statistics();

st.Show();

}

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

{

userBindingSource.DataSource = parent.ctrl.GetUsers();

}

private void dataGridView1\_CellValueChanged(object sender, DataGridViewCellEventArgs e)

{

if(e.RowIndex>-1)

{

var u = (User)userBindingSource[e.RowIndex];

parent.ctrl.Update(u);

}

}

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

{

if(name.Text != parent.user.Name)

{

button4.Visible = true;

}

else

{

button4.Visible = false;

}

}

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

{

if (email.Text != parent.user.Email)

{

button4.Visible = true;

}

else

{

button4.Visible = false;

}

}

private void button4\_Click(object sender, EventArgs e)

{

parent.user.Name = name.Text;

parent.user.Email = email.Text;

parent.ctrl.Update(parent.user);

button4.Visible = false;

}

private void dataGridView2\_CellClick(object sender, DataGridViewCellEventArgs e)

{

if (e.RowIndex > -1)

{

chBookForm = new ChangeBook(books[e.RowIndex], this);

chBookForm.Show();

}

}

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

{

userBindingSource.DataSource = parent.ctrl.GetUsers().FindAll(x => x.Email.StartsWith(textBox1.Text));

}

private void button5\_Click(object sender, EventArgs e)

{

textBox1.Text = "";

userBindingSource.DataSource = parent.ctrl.GetUsers();

}

}

}

**RegForm.cs**

using KR.Controllers;

using KR.Models;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace KR

{

public partial class RegForm : Form

{

public UserForm parent { get; set; }

public RegForm()

{

InitializeComponent();

}

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

{

if(textBox1.Text == "" || textBox2.Text == "" || textBox3.Text == "")

{

note.Visible = true;

return;

}

User user = new User

{

UserId = parent.parent.ctrl.GetNewUserId(),

Name = textBox1.Text,

Email = textBox2.Text,

Password = textBox3.Text,

IsAdmin = false

};

parent.parent.ctrl.Add(user);

parent.CloseForm();

}

private void clearNote(object sender, EventArgs e)

{

note.Visible = false;

}

}

}

**Reports.cs**

using KR.Controllers;

using KR.Models;

using KR.Services;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace KR

{

public partial class Reports : Form

{

private NewReport addForm;

public Form1 parent;

private List<Report> reports;

public Reports(Form1 p)

{

InitializeComponent();

parent = p;

foreach (Report r in parent.ctrl.GetReports())

{

this.ShowReport(r);

}

Button b = new Button();

b.Text = "Добавить отзыв";

b.Width = 200;

b.Height = 40;

b.Margin = new Padding(3, 14, 3, 3);

b.Click += new System.EventHandler(ShowForm);

this.p.Controls.Add(b);

}

private void ShowReport(Report r)

{

GroupBox rr = new GroupBox();

rr.Height = 200;

rr.Text = r.User.Name;

Label car = new Label();

car.Dock = DockStyle.Bottom;

Label date = new Label();

date.Dock = DockStyle.Bottom;

RichTextBox l = new RichTextBox();

l.BorderStyle = BorderStyle.None;

l.Multiline = true;

l.Dock = DockStyle.Fill;

l.ReadOnly = true;

l.Font = new Font("Microsoft Sans Serif", 12);

date.Font = new Font("Microsoft Sans Serif", 10);

car.Font = new Font("Microsoft Sans Serif", 10);

l.Text = r.Text;

date.Text = "Дата: " + r.CreateDate.ToShortDateString();

car.Text = "Автомобиль: " + r.Car.Model;

rr.Controls.Add(l);

rr.Controls.Add(date);

rr.Controls.Add(car);

p.Controls.Add(rr);

}

private void ShowForm(object sender, EventArgs e)

{

if(parent.user == null)

{

parent.showForm();

}

else

{

addForm = new NewReport(this);

addForm.parent = this;

addForm.Show();

}

}

public void CloseForm()

{

addForm.Hide();

}

public void AddReport(Report r)

{

r.User = parent.user;

parent.ctrl.Add(r);

reports = parent.ctrl.GetReports();

p.Controls.Remove(p.Controls[p.Controls.Count - 1]);

ShowReport(reports[reports.Count - 1]);

Button b = new Button();

b.Text = "Добавить отзыв";

b.Width = 200;

b.Height = 40;

b.Margin = new Padding(3, 14, 3, 3);

b.Click += new System.EventHandler(ShowForm);

p.Controls.Add(b);

}

}

}

**Statistics.cs**

using KR.Controllers;

using KR.Models;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace KR

{

public partial class Statistics : Form

{

private RentCarController \_ctrl = new RentCarController();

private List<Car> \_cars;

private List<Book> \_books;

private List<Profit> \_profits;

private List<CarRating> \_ratingCars;

public Statistics()

{

InitializeComponent();

}

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

{

dateTimePicker1.Value = new DateTime(DateTime.Now.Year, DateTime.Now.Month, 1);

dateTimePicker2.Value = new DateTime(DateTime.Now.Year, DateTime.Now.Month+1, 1);

\_cars = \_ctrl.GetCars();

\_ratingCars = \_cars.Select(x => new CarRating { Model = x.Model, ReportsNumber = \_ctrl.GetCarReports(x.CarId).Count, Mark = GetCarMark(x.CarId) }).OrderByDescending(x => x.Mark).ToList();

for(int i = 0; i<\_ratingCars.Count; i++)

{

\_ratingCars[i].Id = i + 1;

}

carRatingBindingSource.DataSource = \_ratingCars;

\_books = \_ctrl.GetBooks();

ReloadProfits();

}

private void ReloadProfits(object sender = null, EventArgs e = null)

{

if(\_books != null)

{

if (dateTimePicker2.Value < dateTimePicker1.Value)

{

var t = dateTimePicker2.Value;

dateTimePicker2.Value = dateTimePicker1.Value;

dateTimePicker1.Value = t;

}

\_profits = \_books.FindAll(x => x.DateStart < dateTimePicker2.Value && x.DateStart > dateTimePicker1.Value).Select(x => new Profit { Id = x.BookId, ClientName = x.User.Name, DateFinish = x.DateFinish, DateStart = x.DateStart, Model = x.Car.Model, Sum = x.Car.Price \* (x.DateFinish - x.DateStart).Days }).ToList();

profitBindingSource.DataSource = \_profits;

label3.Text = \_ctrl.GetB().Where(x => x.DateStart < dateTimePicker2.Value && x.DateStart > dateTimePicker1.Value).Select(x => x.Car.Price \* (x.DateFinish - x.DateStart).Days).Sum().ToString();

}

}

private void carRatingBindingSource\_CurrentChanged(object sender, EventArgs e)

{

}

private void groupBox4\_Enter(object sender, EventArgs e)

{

}

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

{

}

private double GetCarMark(int CarId)

{

var rr = \_ctrl.GetCarMarks(CarId);

if(rr.Count > 0)

{

return rr.Average();

}

else

{

return 0;

}

}

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

{

\_ctrl.Export(\_ratingCars);

}

}

}

**UserForm.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace KR

{

public partial class UserForm : Form

{

public Form1 parent { get; set; }

private RegForm regForm { get; set; }

public UserForm()

{

InitializeComponent();

}

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

{

regForm = new RegForm();

regForm.parent = this;

regForm.Show();

}

public void CloseForm()

{

regForm.Hide();

}

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

{

parent.EnterUser(textBox1.Text, textBox2.Text, checkBox1.Checked);

}

}

}

**Program.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace KR

{

static class Program

{

/// <summary>

/// The main entry point for the application.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Form1 f1 = new Form1();

Application.Run(f1);

}

}

}