Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Базы данных знаний и экспертные системы (БДЗиЭС)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему:

ВЕБ-ПРИЛОЖЕНИЕ ДЛЯ АГРЕГАТОРА ОБЪЯВЛЕНИЙ О ПРОДАЖЕ АВТОМОБИЛЕЙ

БГУИР КП 1-40 01 01  014  ПЗ

Студент: гр. 651001 Лащиновская К.Д.

Руководитель: асс. Фадеева Е.Е.

Минск 2019

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ

Заведующий кафедрой ПОИТ

––––––––––––––––––––––––

(подпись)

Лапицкая Н.В. 2019 г.

ЗАДАНИЕ

по курсовому проектированию

Студенту    –––––––––––––

1. Тема работы    ––––

2. Срок сдачи студентом законченной работы 20.12.2019

3. Исходные данные к работе   .

4. Содержание расчётно-пояснительной записки (перечень вопросов, которые подлежат разработке)

*Введение.*

*1.Аналитический обзор литературы и существующих аналогов;*

*2.Разработка функциональных требований к программному стредству;*

*3. Разработка программного средства;*

*4. Разработка базы данных;*

*5. Тестирование, экспериментальные исследования и анализ полученных результатов;*

*6. Руководство пользователя программы;*

*Заключение, список литературы, ведомость, приложения.*

5. Перечень графического материала (с точным обозначением обязательных чертежей и графиков)

1. Схема программы

6. Консультант по курсовому проекту Фадеева Е.Е.

7. Дата выдачи задания 24.10.2019 г.––   –

8. Календарный график работы над проектом на весь период проектирования (с обозначением сроков выполнения и процентом от общего объёма работы):

раздел 1, введение к 30.10.2019 – 10 % готовности работы;

разделы 2 к 10.11.2019 – 30 % готовности работы;

разделы 3,4 к 25.11.2019 – 60 % готовности работы;

раздел 5, 6 к 15.12.2019 – 90 % готовности работы;

оформление пояснительной записки и графического материала к 20.12.2019 – 100 % готовности работы.

Защита курсового проекта с 15.12 по 24.12 2019 г.

РУКОВОДИТЕЛЬ Е.Е. Фадеева

(подпись)

Задание принял к исполнению –––\_\_\_\_––     24.10.2019 г.

(дата и подпись студента)

**СОДЕРЖАНИЕ**

ВВЕДЕНИЕ 5

1 АНАЛИТИЧЕСКИЙ ОБЗОР ЛИТЕРАТУРЫ И

СУЩЕСТВУЮЩИХ АНАЛОГОВ 6

[1.1 Аналитический обзор литературы 6](#_Toc484199539)

1.2 Обзор существующих аналогов 7

[1.3 Постановка задачи на курсовое проектирование 10](#_Toc484199539)

[2 РАЗРАБОТКА ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ К ПРОГРАММНОМУ СРЕДСТВУ 11](#_Toc484199539)

[2.1 Спецификация функциональных требований 11](#_Toc484199539)

[2.2 Пользователи системы и их роли 12](#_Toc484199539)

[3 РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА 14](#_Toc484199539)

[3.1 Разработка интерфейса приложения 14](#_Toc484199539)

[3.2 Разработка серверной части приложения 14](#_Toc484199539)

3.3 Назначение пользовательских классов 15

4 РАЗРАБОТКА БАЗЫ ДАННЫХ 16

[4.1 Обоснование выбора СУБД и библиотеки для работы с базой данных 16](#_Toc484199539)

[4.2 Сущности и связи базы данных 17](#_Toc484199539)

[4.3 Назначение основных процедур, функций и представлений базы данных 19](#_Toc484199539)

[5 ТЕСТИРОВАНИЕ 20](#_Toc484199539)

[6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 23](#_Toc484199539)

[6.1 Инструкция по использованию приложения 23](#_Toc484199539)

[6.2 Указания по установке 25](#_Toc484199539)

[ЗАКЛЮЧЕНИЕ 26](#_Toc484199539)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 27](#_Toc484199539)

[Приложение А Текст программного модуля главного графического компонента приложения 28](#_Toc484199539)

[Приложение Б Текст программного модуля контроллера приложения 30](#_Toc484199539)

[Приложение В Текст программного модуля сервиса приложения 31](#_Toc484199539)

[Приложение Г Дамп структуры базы данных 35](#_Toc484199539)

**ВВЕДЕНИЕ**

На сегодняшний день широко распространены Интернет-площадки для размещения объявлений о продаже или покупке недвижимости, автомбилей, предметов быта и т.д. Данные ресурсы позволяют охватить более широкую аудиторию, повысить скорость отклика на объявление. Зачастую лица, заинтересованные в приобретении какого-либо товара, используют несколько ресурсов для поиска оптимального варианта для покупки. Изучение объявлений на нескольких сайтах занимает достаточно большое время. В связи с этим большую популярность приобретают веб-приложения, агрегирующие информацию с нескольких сайтов и позволяющие одним поисковым запросом охватить объявления сразу из нескольких ресурсов.

В данном курсовом проекте производится разработка веб-приложения для агрегации объявлений о продаже автомобилей.

При разработке подобных приложений большое внимание уделяется скорости поиска объявлений по полученному от пользователя запросу, гибкости базы данных для получения необходимых результатов, а также удобству и понятности пользовательского интерфейса. Также при создании современных веб-сайтов большое внимание уделяется адаптивной верстке, так как пользователи очень часто используют веб-приложения на мобильных устройствах и планшетах.

Пояснительная записка к курсовому проекту состоит из шести разделов. Первый раздел посвящен рассмотрению существующих аналогов разрабатываемого программного средства. В данном разделе сформулирована задача на курсовое проектирование. Во втором разделе проводится анализ требований к программному средству и разработка функциональных требований. В третьем разделе приводится пошаговый процесс разработки логики работы программного средства и пользовательского интерфейса, даны описания основных классов приложения и их методов, обосновано использование тех или иных типов данных. Четвертый раздел посвящен разработке базы данных для программного средства. В пятом разделе приведен набор тестов для проектируемого программного средства. Особое внимание уделено проверке работы программы в критических ситуациях (отсутствие Интернет-соединения, некорректность аутентификационных данных). Шестой раздел представляет собой руководство пользователя данного программного средства.

**1 АНАЛИТИЧЕСКИЙ ОБЗОР ЛИТЕРАТУРЫ И СУЩЕСТВУЮЩИХ**

**АНАЛОГОВ**

* 1. **Аналитический обзор литературы**

Агрегаторы объявлений - это сайты, которые собирают рекламные объявления различных компаний и позволяют их фильтровать по различным критериям [1]. Как правило, такие ресурсы позволяют просматривать предложения как от частных лиц, так и от коммерческих организаций. Такой формат гораздо удобнее, чем посещение каждого сервиса по отдельности, ведь они агрегируют предложения с самых популярных досок или выбранных пользователем сайтов по заданным параметрам на одной странице. Такие ресурсы экономят время как покупателю, так и продавцу товаров или услуг.

Некоторые из сайтов агрегаторов имеют свои особенности, но в общем их объединяют такие функции:

* сбор объявлений заданной тематики со всех доступных или выбранных досок;
* автоматическое размещение объявлений на разных площадках;
* фильтры поиска по темам, подразделам, наименованиям товаров/услуг, ценам, локации и многим другим параметрам;
* рассылка и оповещение (при желании) о появлении новых объявлений.

В зависимости от правил конкретного сервиса можно использовать его без регистрации или зарегистрироваться и получить доступ к расширенным функциям: закладкам, рассылке и т.д. Размещение объявлений требует регистрации, а во многих случаях и оплаты, простой просмотр объявлений обычно бесплатен.

Как и большинство современных веб-приложений, электронные системы бронирования представляют собой одностраничные приложения с REST-архитектурой.

* + 1. REST-архитектура

REST (сокращение от англ. Representational State Transfer — «передача состояния представления») — архитектурный стиль взаимодействия компонентов распределённого приложения в сети [2]. REST представляет собой согласованный набор ограничений, учитываемых при проектировании распределённой гипермедиа-системы. В определённых случаях (интернет-магазины, поисковые системы, прочие системы, основанные на данных) это приводит к повышению производительности и упрощению архитектуры. В широком смысле компоненты в REST взаимодействуют наподобие взаимодействия клиентов и серверов во Всемирной паутине.

В отличие от веб-сервисов (веб-служб) на основе SOAP, не существует «официального» стандарта для RESTful веб-API. Дело в том, что REST является архитектурным стилем, в то время как SOAP является протоколом. Несмотря на то, что REST не является стандартом сам по себе, большинство RESTful-реализаций используют стандарты, такие как HTTP, URL, JSON и XML.

Свойства архитектуры, которые зависят от ограничений, наложенных на REST-системы:

* производительность — взаимодействие компонентов системы может являться доминирующим фактором производительности и эффективности сети с точки зрения пользователя;
* масштабируемость для обеспечения большого числа компонентов и взаимодействий компонентов.

Архитектура REST влияет на масштабируемость следующим образом:

* простота унифицированного интерфейса;
* открытость компонентов к возможным изменениям для удовлетворения изменяющихся потребностей (даже при работающем приложении);
* прозрачность связей между компонентами системы для сервисных служб;
* переносимость компонентов системы путем перемещения программного кода вместе с данными;
* надёжность, выражающаяся в устойчивости к отказам на уровне системы при наличии отказов отдельных компонентов, соединений или данных.

Существует шесть обязательных ограничений для построения распределённых REST-приложений:

1. модель клиент-сервер
2. отсутствие состояния (в период между запросами клиента никакая информация о состоянии клиента на сервере не хранится);
3. кэширование (клиенты, а также промежуточные узлы, могут выполнять кэширование ответов сервера);
4. единообразие интерфейса (унифицированные интерфейсы позволяют каждому из сервисов развиваться независимо);
5. слои (применение промежуточных серверов способно повысить масштабируемость за счёт балансировки нагрузки и распределённого кэширования);
6. код по требованию (расширение функциональности клиента за счёт загрузки кода с сервера в виде апплетов или сценариев).
   1. **Обзор существующих аналогов**
      1. Haraba.ru

Haraba.ru – агрегатор автомобильных объявлений с сайтов во всех регионах России. Данное приложение анализирует объявления с таких ресурсов, как «Avito», «auto.ru», «Юла», «drom.ru». В приложении используется политика тарифных планов: в зависимости от выбранного тарифа пользователю становятся доступны некоторые возможности приложения, меняется частота обновления информации, варьируется количество возможных переходов по объявлениям в день и количество проверок на некорректные объявления. Найденные объявления форматируются при помощи различных цветов и условных обозначений, что ускоряет навигацию пользователя по списку объявлений [3].

К основным *достоинствам* данного ресурса относятся:

* гибкий фильтр для поиска объявлений, возможность осуществлять поиск по отдельным словам в тексте объявления;
* возможность настроить уведомления о новых объявлениях по электронной почте или при помощи мессенджера Telegram;
* возможность просматривать историю изменения цены автомобиля;
* статистика отклонений цен на автомобили от рыночной цены.

К *недостаткам* ресурса можно отнести следующее:

* поиск объявлений доступен только зарегистрированным пользователям;
* мобильное приложение ресурса доступно только для ОС Android;
* отсутствие какого-либо дополнительного функционала для бесплатного тарифа (доступен только поиск с задержкой информации на 3 часа).

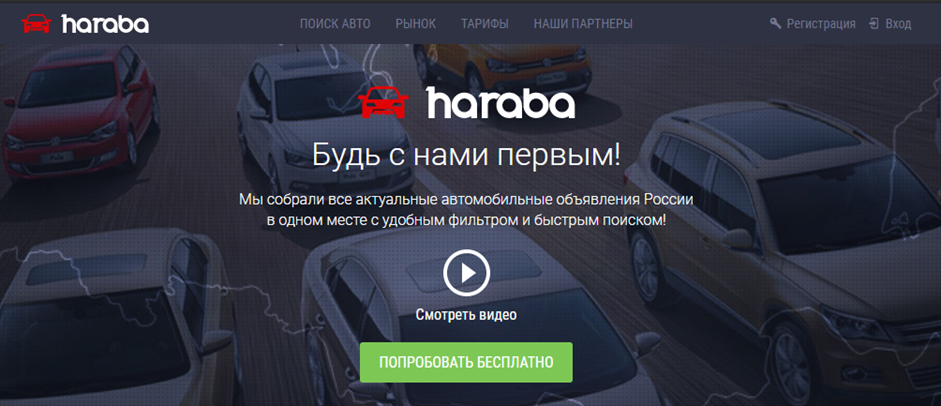


Рисунок 1.1 - Главная страница приложения Haraba.ru

* + 1. Robasta.ru

Robasta.ru – агрегатор объявлений, позволяющий искать новые и б/у автомобили по заниженным ценам. В приложении доступен поиск с множеством дополнительных фильтров, при отображении результатов поиска выводится диаграмма, отражающая разницу между актуальной и рыночной стоимостью автомобилей. Данный сервис сортирует автомобили по выгодности от их приобретения, учитывая все характеристики транспортного средства. При размещении объявления о продаже существует возможность получить отчет об оценке автомобиля [4].

К основным *достоинствам* сервиса можно отнести:

* возможность отслеживать свежие объявления о продаже б/у автомобилей в режиме реального времени;
* поиск автомобилей доступен для незарегистрированных пользователей;
* возможность получать уведомления о новых объявлениях через мессенджер Telegram;
* наличие API для сторонних разработчиков.

*Недостатком* сервиса является отсутствие мобильной версии приложения. Однако наличие API позволяет программистам делать надстройки над данным сервисом и предоставлять отдельное мобильное приложение или приложение с расширенным функционалом.

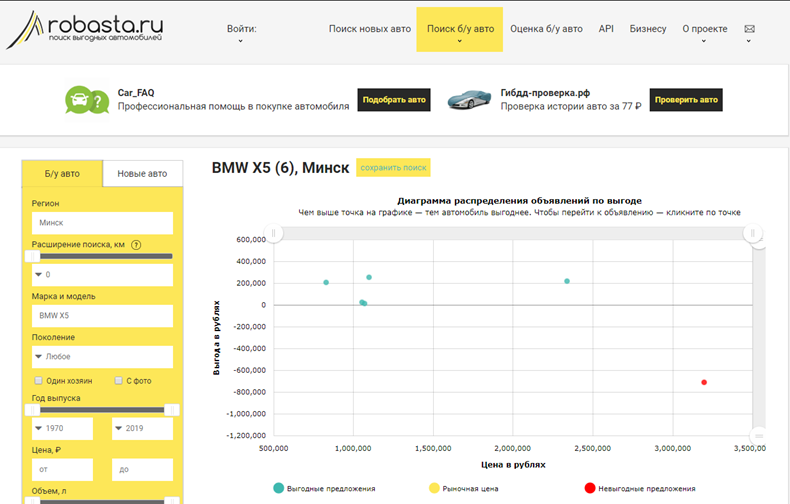


Рисунок 1.2 - Главная страница приложения Robasta.ru

* + 1. Autobot.by

Autobot.by (АвтоБОТ) - система для поиска объявлений о продаже автомобилей на территории Республики Беларусь [5].

К *достоинствам* сервиса можно отнести следующее:

* небольшая задержка обновления информации (около 10 минут);
* возможность получения любых данных сервиса при помощи API;
* возможность настраивать фильтры поиска и получать по ним уведомления по электронной почте, в Telegram и Viber.

*Недостатками* сервиса являются недостаточно понятный пользовательский интерфейс, большое количество рекламы на сайте и отсутствие мобильной версии сайта.

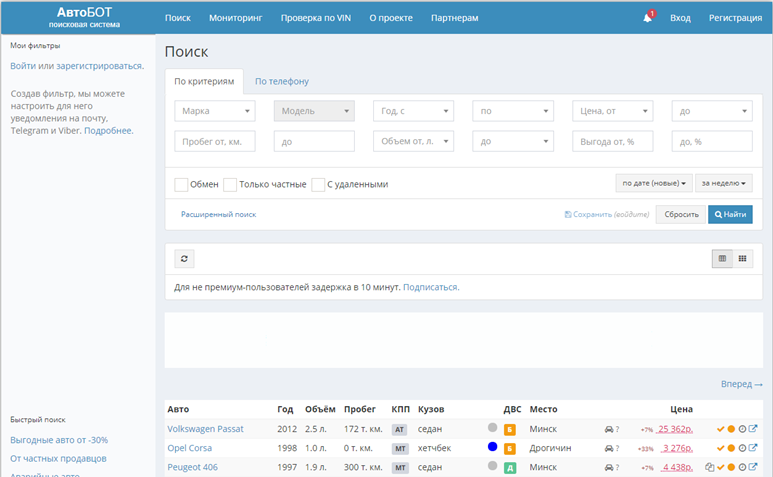


Рисунок 1.3 - Главная страница приложения Autobot.by

* 1. **Постановка задачи на курсовое проектирование**

Рассмотрев возможные аналоги программных средств и изучив особенности REST-архитектуры, можно сформулировать задачу на курсовое проектирование.

Необходимо разработать веб-приложение для агрегации авто-объявлений с различных ресурсов, имеющее следующий функционал:

* поиск объявлений по различным параметрам;
* добавление объявлений в личный кабинет;
* возможность просмотра истории поисковых запросов;
* статистика по запросам на сайте.

Входные данные: аутентификационные данные пользователей, параметры поиска объявлений.

Выходные данные: результаты поиска, статистика запросов.

Данные объявлений хранятся в формате Xml.

Условия написания: серверная часть приложения должна быть написана на .NET CORE 2.1 по принципу WebAPI, клиентская часть должна быть написана с использованием JavaScript-библиотеки Vue. Разрабатываемое приложение должно быть одностраничным, данные между клиентом и сервером передаются в формате JSON. В качестве СУБД использовать MS SQL Server 2012.

**2 РАЗРАБОТКА ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ К ПРОГРАММНОМУ СРЕДСТВУ**

**2.1 Спецификация функциональных требований**

Проанализировав задачу на курсовое проектирования, можно сформулировать следующие функциональные требования:

1. При открытии приложения должна отображаться страница с формой авторизации и кнопками «Войти», «Регистрация» и «Продолжить без авторизации».
2. На странице регистрации должны быть следующие поля: «Email», «Логин», «Пароль», «Повторите пароль». При успешной регистрации пользователь переходит на страницу авторизации. Если в ходе регистрации произошла ошибка, должно отображаться сообщение об ошибке.
3. После прохождения авторизации (либо при нажатии кнопки «пройти без авторизации») пользователь переходит на главную страницу.
4. Если пользователь прошел авторизацию, то в верхней части страницы отображаются кнопки «Личный кабинет”, «Выйти” и «Настройки”. При нажатии на кнопку «Личный кабинет” осуществляется переход на страницу пользователя. При нажатии на кнопку «Выйти” осуществляется выход из приложения, и пользователь возвращается на страницу авторизации. При нажатии на кнопку «Настройки” осуществляется переход на страницу настроек приложения для данного пользователя.
5. Если пользователь не прошел авторизацию, то кнопки из п. 4 отсутствуют.
6. На главной странице должна отображаться форма для ввода поискового запроса. Поиск автомобилей производится по следующим критериям: Интернет-ресурс, марка, модель, тип двигателя, тип кузова, состояние машины:

* критерий «Интернет-ресурс» является обязательным для заполнения и имеет множественный выбор;
* должен быть обязательным для заполнения хотя бы один из следующих критериев: марка, тип двигателя, тип кузова, состояние машины;
* критерии марка, модель, тип двигателя, тип кузова, состояние машины должны иметь только одиночный выбор;
* в элементе выбора марки машины должен быть разрешен поиск по названию, при вводе последовательности символов осуществляется мгновенный поиск подходящих марок машин;
* если критерии поиска заполнены неверно, поиск не производится, на экране появляется сообщение об ошибке.

1. При вводе параметров поиска и нажатии на кнопку «Найти» производится поиск объявлений. Найденные объявления выводятся ниже формы поиска.
2. Список найденных объявлений должен позволять добавить выбранное объявления в «Отслеживаемые объявления»
3. На странице «Личный кабинет» в зависимости от роли пользователя должны быть три вкладки: «Отслеживаемые объявления», «История поиска», «Аналитика».
4. Во вкладке «Отслеживаемые объявления» должны отображаться те объявления, которые пользователь добавил в «отслеживаемые» на странице поиска (главная страница):

* на каждом объявлении должна быть кнопка «Удалить» для удаления объявления из отслеживаемых объявлений;
* при открытии вкладки должна обновляться информация во всех объявлениях.

1. Во вкладке «История поиска» должны отображаться последние 5 поисковых запросов, которые задавал пользователь:

* для каждого запроса должен отображаться сайт поиска, и другие критерии, присутствовавшие в запросе; если какой-либо критерий отсутствовал в запросе, то он не должен отображаться в информации о запросе.

1. Во вкладке «Аналитика» должна быть представлена аналитика по поисковым запросам от пользователей:

* должна быть представлена аналитика по следующим параметрам: марки автомобилей, состояние автомобилей;
* аналитика должна быть представлена в виде круговых диаграмм;
* на диаграмме должна быть возможность временно убрать некоторые элементы выборки.

1. На странице «Настройки» должна быть возможность включить «Pro»-режим. Если пользователь уже работает в данном режиме, на странице «Настройки», должно отображаться соответствующее сообщения, кнопка должна отсутствовать.

Данные в программу поступают из двух источников: пользователи и ресурсы с объявлениями о продаже автомобилей. Пользователи вводят свои аутентификационные данные и поисковые запросы. Через равные промежутки времени программа запрашивает данные от парсера и производится запись полученных данных в таблицу базы данных. При обработке запросов от пользователей осуществляется обращение к базе данных, содержащей результаты работы парсера. При обновлении данных о сохраненных объявлениях также производится обращение к базе. При вводе поискового запроса он сохраняется в базу данных (для авторизованных пользователей). Результатом работы веб-приложения являются найденные объявления и графики, отражающие статистику поисковых запросов.

**2.2** **Пользователи системы и их роли**

Предполагается, что разрабатываемое приложение будет доступно широкому кругу пользователей в сети Интернет.

В приложении предполагается четыре роли:

* гость;
* зарегистрированный пользователь;
* зарегистрированный пользователь с «Pro»-аккаунтом;
* администратор.

Гость может только производить поиск по объявлениям. Зарегистрированный пользователь имеет возможность производить поиск по объявлениям и просматривать историю своих поисковых запросов для более быстрого повторного ввода запроса. Пользователь с «Pro»-аккаунтом имеет все возможности обычного зарегистрированного пользователя, дополнительно он может добавлять объявления в отслеживаемые, добавленные объявления будут обновляться, когда пользователь заходит в личный кабинет. Данный тип аккаунта включается в настройках сайта (страница «Настройки»). Администратор имеет все возможности обычного зарегистрированного пользователя, дополнительно он может просматривать статистику поисковых запросов от других пользователей. Изначально администратор не имеет возможностей «Pro»-пользователя. Данная возможность подключается тем же способом, что и для обычного пользователя.



Рисунок 2.1 – UseCase диаграмма разрабатываемого программнго средства

**3 РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА**

* 1. **Разработка интерфейса приложения**

Для создания клиентской части приложения использовалась JavaScript-библиотека Vue. Данная библиотека позволяет создавать одностраничные приложения.

В основу построения приложения была положена модульная архитектура, при которой создаются компоненты, которые затем можно заново использовать, меняя только некоторые параметры. Все визуальные компоненты были разделены на два типа: «умные» компоненты (контейнеры) и «глупые» компоненты. К контейнерам можно отнести следующие компоненты: страница регистрации (LoginPage), страница поиска (MainPage) и др. Данные компоненты осуществляют получение данных, передают полученные данные «глупым» компонентам и управляют собственным состоянием. «Глупые» компоненты не осуществляют получение данных, а лишь отображают данные, полученные через свойства (props). К «глупым» компонентам можно отнести кнопки (Button), заголовок (Header) и др. Также при помощи пакетного менеджера npm в приложение были внедрены готовые компоненты, имеющие необходимый функционал.

Для хранения и управления глобальным состоянием приложения использовалась библиотека Vuex. Данная библиотека может использоваться для сохранения информации об авторизованном пользователе, параметрах навигации [6].

Для обращения к API приложения использовалась библиотека axios. При помощи данной библиотеки можно можно осуществлять общение с сервером через JSON.

* 1. **Разработка серверной части приложения**

Серверная часть приложения написана с использованием ASP .NET Core WebAPI. При данном подходе приложение представляет собой веб-службу, которая имеет свой API и полностью не зависит от клиентской части. Таким образом, для одного серверного приложения можно создать большое количество различных клиентских приложений [7].

Разработанное приложение построено по трехуровневой архитектуре. Первоначально происходит обращение к контроллеру, в котором осуществляется получение запросов, извлечение данных из запросов и отправка ответа сервера. Контроллеры обращаются к сервисам, в которых содержится вся бизнес-логика приложения, реализуются все алгоритмы. Для получения информации из базы данных сервисы используют провайдеры, классы, предназначенные для работы с базой данных [8].

* 1. **Назначение пользовательских классов**

Таблица 3.1 - Назначение пользовательских классов

|  |  |
| --- | --- |
| Имя класса | Назначение |
| UsersController | Класс для приема запросов на регистрацию и авторизацию от клиентского приложения. Также используется для приема запросов на изменение типа учетной записи пользователя. |
| SearchController | Класс для приема запросов от приложения-клиента, связанных с поиском объявлений, историей поиска. |
| CarSubscriptionController | Класс для приема запросов от приложения-клиента, связанных с подпиской пользователя на объявления, отпиской от объявлений и выдаче объявлений, на которые подписан пользователь. |
| UserService | Класс, содержащий бизнес-логику по работе с данными пользователей. |
| SearchService | Класс, содержащий бизнес-логику для работы с поисковыми запросами и историей поиска. |
| CarService | Класс, содержащий бизнес-логику для работы с подписками на объявления. |
| UserProvider | Класс, выполняющий запросы к базе данных, связанных с данными пользователей. |
| SearchProvider | Класс, выполняющий запросы к базе данных, связанных с поиском данных и историей поиска. |
| CarProvider | Класс, выполняющий запросы к базе данных, связанных с подпиской на объявления об автомобилях. |
| Analitics, Search, User, BaseSearchParameters | Классы для передачи и приема данных между различными слоями приложения, а также между клиентским и серверным приложением. |
| XmlParser | Класс для получения информации из xml-файла с данными об объявлениях. |

**4 РАЗРАБОТКА БАЗЫ ДАННЫХ**

* 1. **Обоснование выбора СУБД и библиотеки для работы с базой данных**

В качестве СУБД для проектируемого приложения была выбрана MS SQL Server 2012. SQL Server содержит процессор запросов, который обеспечивает поддержку баз данных очень большого объема и обработку сложных запросов. Среди его особенностей – использование составных индексов, новые алгоритмы хеширования и слияния, множественные триггеры, а также обработка гетерогенных, распределенных и параллельных запросов. Увеличенный до 8 КБ размер страниц способствует быстрому извлечению данных, позволяет использовать строки и столбцы большего размера, что открывает возможность эффективного хранения сложных, подробных данных. Менеджер блокировок динамически адаптирует алгоритм использования ресурсов в больших базах данных, что делает продукт наиболее пригодным для интерактивной обработки транзакций (online transaction processing - OLTP) и создания хранилищ данных. SQL Server поддерживает лингвистический поиск, позволяя создавать специальные индексы ключевых слов или фраз для выбранных столбцов или таблиц. Также SQL Server хорошо интегрируется с веб-приложениями, написанными на платформе .NET Framework [9].

Для взаимодействия веб-приложения с базой данных была выбрана библиотека ADO.NET. Библиотека ADO.NET создана для использования в управляемых (managed) проектах. При использовании DataSet класса библиотеки работа происходит с отсоединенными наборами данных. Это означает, что вы используете соединение с источником данных очень короткое время. Во многих системах количество подключений к базам данных является самым узким местом в плане масштабируемости. И для этих систем ADO.NET является очень хорошим решением, резко повышающим их масштабируемость. Отключенный набор данных может использоваться несколькими частями программы (или пользователями) одновременно. В ADO.NET DataSet действительно независим от источника данных, и изменение провайдера, с помощью которого заполняется DataSet, не влияет на функциональность DataSet-а. А то, что данные читаются целиком или последовательно, снижает планку требований, предъявляемых к провайдеру. Так что в большинстве случаев изменение кода будет состоять в изменении используемого управляемого провайдера данных и строки подключения. Так как ADO.NET использует XML как стандартный формат передачи данных, программа, которой необходимо получить данные из компонента ADO.NET, не обязана сама быть компонентом ADO.NET. В общем случае она вообще может не быть Windows-программой. Единственное требование – эта программа должна понимать XML. И это позволяет ADO.NET-компонентам при использовании других компонентов и сервисов, входящих в VS.Net, легко взаимодействовать с любой программой на любой платформе [10].

* 1. **Сущности и связи базы данных**

Основными сущностями базы данных являются сущности «User», «Car» и «Search». Сущность User представляет информацию о пользователе приложения. Сущность Car представляет информацию об автомобиле, используемом в системе (под автомобилем в системе подразумевается объявление о продаже соответствующего автомобиля). Сущность Search представляет информацию о поисквом запросе пользователя. Также в базе данных существуют сущности, представляющие различные характеристики автомобилей, по которым в дальнейшем будет производиться поиск. К таким сущностям относятся: BodyType (тип кузова автомобиля), EngineType (тип двигателя), Condition (состояние автомобиля), SellerType (тип прдавца автомобиля) и др. В базе данных существует сущность ErrorLog, которая содержит информацию об ошибках, произошедших при выполнении различных операций в системе.

Таблицы, содержащие характеристики автомобилей являются таблицами-справочниками. Таблица Car ссылается на справочные таблицы. Таблицы, содержащие данные об ошибках и настройки разбора объявлений являются независимыми таблицами и не имеют связей с другими таблицами. Взаимодействие с данными таблицами (вставка и изменение значений) осуществляется внутри хранимых процедур а также в коде веб-приложения. Подписка пользователей на объявления об автомобилях представляется в виде связи многие-ко-многим. Для разрешения данной связи была введена дополнительная таблица UserCarSubscription, содержащая идентификаторы пользователей и автомобилей. Информация о роли пользователя содержится в тдельной таблице, UserSubscription, которая ссылается на таблицу пользователей User.

Ниже представлена модель базы данных (см. рисунок 4.1).



Рисунок 4.1 - Модель базы данных приложения

* 1. **Назначение основных процедур, функций и представлений базы данных**

Таблица 4.1 - Назначение объектов базы данных

|  |  |
| --- | --- |
| Имя объяета | Назначение |
| dbo.CarParsingInsert (хранимая процедура) | Вставка разобранных данных из объявления в промежуточную таблицу dbo.CarParsing для последующего поиска. |
| dbo.CarParsingMergeCar (хранимая процедура) | Вставка данных об объявлении из промежуточной таблицы dbo.CarParsing в таблицу dbo.Car (используется при добавлении пользователем объявления в «отслеживаемые»). |
| dbo.CarOwnerPhoneInsert (хранимая процедура) | Вставка телефонного номера владельца автомобиля в таблицу. |
| dbo.CarLogInsert (хранимая процедура) | Вставка сообщения при логировании операций в системе. |
| dbo.CarParsingSettingsGet (хранимая процедура) | Получение настроек для парсера объявлений с авто-сайтов. |
| dbo.CarUpdate\_IsActive (хранимая процедура) | Изменение статуса автомбиля (активный либо не автивный) |
| dbo.CurrencyRateGet (хранимая процедура) | Получение данных о курсах валют за определенный период. Процедура выполняет запрос (через процедуру dbo.HTTPCall) к Интернет-ресурсу, выдающему информацию о валютах. |
| dbo.MappingType (функция) | Перевод словестного представления характеристики автомобиля в численное представление для установки идентификатора значения характеристики в таблице dbo.Car. |
| dbo.ParseJSON (функция) | Разбор JSON-файла. |
| dbo.XMLToJSON (функция) | Конвертация XML содержимого в JSON-формат. |
| dbo.vwCarInfo (представление) | Представление полной информации об автомобиле, характеристики автомобиля представляются в словестном виде. |
| dbo.vwCarActive (представление) | Представление всех активных авто-объявлений в списке. |
| dbo.vwCarBrandModel (представление) | Представление всех возможных моделей автомобилей с указанием марки автомобиля (одна и та же марка автомобиля встречается в одной или нескольких записях в таблице представления). |

**5** **ТЕСТИРОВАНИЕ**

Тестирование программного средства было проведено путем ввода различных параметров поиска и добавления/удаления объявлений из списка «отслеживаемых». Также большое значение уделено тестированию функционала для администратора и разграничению прав доступа на просмотр различных страниц.

Тестирование проводилось по следующим группам тестов:

* тестирование аутентификации (осуществлялись проверки на проведение аутентификации при различных состояниях сети);
* тестирование отображения диаграмм на странице администратора;
* тестирование поиска объявлений по заданным параметрам;
* тестирование добавления объявлений в «отслеживаемые» и удаление из данного списка.

Таблица 5.1 – Тестирование программы при некорректных входных данных

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Тестируемое действие | Ход выполнения | Ожидаемый результат | | Полученный результат | | |
| Аутентификация при отсутствии подключения к сети Интернет | Выполнить вход на сайт. В открывшемся окне ввести имя пользователя и пароль. Нажать кнопку «Войти». | Сообщение «Ошибка сети» | | Тестирование пройдено. | | |
| Аутентификация при некорректных аутентификационных данных | Выполнить вход на сайт. В открывшемся модальном окне ввести имя пользователя и пароль, которых нет в базе данных. Нажать кнопку «Войти». | Сообщение «Неверное имя пользователя или пароль» | | Тестирование пройдено. | | |
| Аутентификация при корректных аутентификационных данных | Выполнить вход на сайт. В открывшемся модальном окне ввести имя пользователя и пароль. Нажать кнопку «Войти». | В шапке сайта появились ссылки на страницу пользователя и настройки. Осуществлен переход на страницу поиска | | Тестирование пройдено. | | |
| Продолжение Таблицы 5.1 | | | | | | |
| Тестируемое действие | Ход выполнения | | Ожидаемый результат | | | Полученный результат |
| Поиск несуществующего автомобиля | Выполнить вход на сайт. Ввести данные в поля для параметров поиска. Нажать кнопку «Поиск». | | Сообщение «Ничего не найдено». | | | Тестирование пройдено. |
| Поиск автомобиля при существовании искомого автомобиля. | Выполнить вход на сайт. В полях для поиска ввести данные. Параметры поиска должны соответствовать существующему автомобилю. Нажать кнопку «Поиск». | | Ниже формы поиска появился список найденных объявлений. | | | Тестирование пройдено. |
| Добавление объявления в «отслеживаемые» | Выполнить вход на сайт. Пройти авторизацию. Выполнить поиск объявлений. Нажать кнопку «Добавить» на одном из найденных объявлений. | | Сообщение «Добавлено». В списке «отслеживаемые» появилось только что добавленное объявление. | | | Тестирование пройдено. |
| Удаление объявления из списка «отслеживаемых» | Выполнить вход на сайт. Пройти авторизацию как администратор. Перейти на страницу личного кабинета. Перейти на вкладку «Отслеживаемые авто». В списке объявлений нажать на кнопку «Удалить» напротив какого-либо объявления. | | Сообщение «Удалено». Объявление не отображается в списке «отслеживаемых» объявлений. | | | Тестирование пройдено. |
|  | | | | | | |
| Продолжение Таблицы 5.1 | | | | | | |
| Тестируемое действие | Ход выполнения | Ожидаемый результат | | | Полученный результат | |
| Включение Pro-аккаунта | Выполнить вход на сайт. Пройти авторизацию как обычный пользователь. Перейти на страницу настроек. Нажать кнопку «Включить Pro-аккаунт». Выйти из приложения путем нажатия на кнопку «Выйти» в шапке сайта. Повторить авторизацию на сайте. | Пользователю стал доступен функционал Pro-аккаунта (добавление объявлений в «отслеживаемые») | | | Тестирование пройдено. | |
| Выход из учетной записи | Выполнить вход на сайт. Пройти авторизацию. В шапке сайта нажать на кнопку «Выйти». | Автоматически произведена навигация на начальную страницу. В шапке сайта не отображаются кнопки перехода на страницу пользователя. Присутствует кнопка «Войти». | | | Тестирование пройдено. | |

**6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ**

* 1. **Инструкция по использованию приложения**
     1. Аутентификация и авторизация

При запуске приложения открывается страница авторизации. Для аутентификации нужно ввести логин и пароль. Далее необходимо нажать кнопку «Войти». Если были введены верные логин и пароль, то пользователь будет авторизован в системе и осуществится переход на главную страницу. Иначе будет выведено сообщение б ошибке. Если пользователю нужно создать новую учетную запись, то в нижней части страницы регистрации необходимо нажать кнопку «Регистрация». В открывшейся форме нужно ввести адрес электронной почты, логин и пароль. Далее необходимо нажать кнопку «Зарегистрироваться». При успешной регистрации пользователь будет перенаправлен на страницу авторизации.

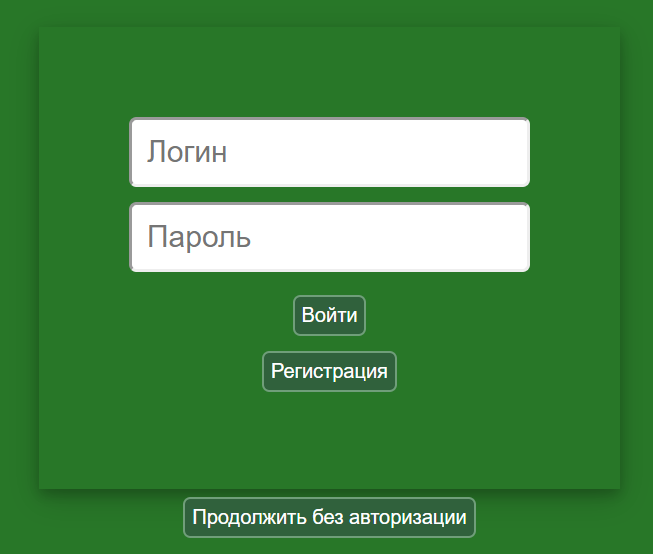


Рисунок 6.1 - Форма для аутентификации

* + 1. Поиск и добавление объявлений в личный список

Сначала необходимо перейти на главную страницу приложения. Далее нужно ввести параметры поиска в форму с заголовком «Поиск», после чего нужно нажать кнопку «Поиск». Далее будет осуществлен поиск, ниже формы поиска появятся найденные объявления. Для того чтобы добавить объявление в «Отслеживаемые» объявления, пользователь должен быть авторизован в системе. На форме найденного объявления в правом верхнем углу необходимо нажать кнопку «Добавить». Добавленное объявление отобразится в списке «Отслеживаемые» в личном кабинете.

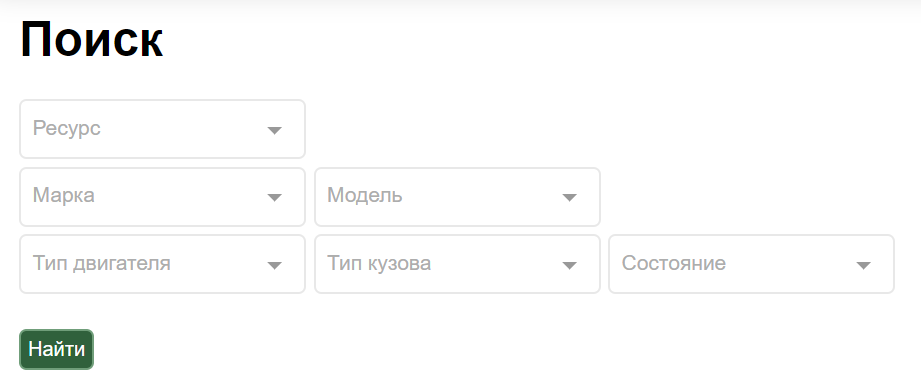


Рисунок 6.2 - Форма для поиска объявления

* + 1. Работа в личном кабинете

При нажатии на кнопку «Личный кабинет» осуществляется переход на страницу «Личный кабинет» (персональная страница пользователя). В зависимости от роли пользователя в личном кабинете доступны три вкладки: «Отслеживаемые объявления», «История поиска», «Аналитика».

Во вкладке «Отслеживаемые объявления» отображаются объявления, которые были добавлены в «отслеживаемые» на главной странице (страница поиска). При нажатии на объявление открывается модальное окно с подробной информацией об автомобиле. В правом верхнем углу объявления находится кнопка «Удалить», при нажатии на которую объявление удаляется из «отслеживаемых».

Во вкладке «История поиска» отображается пять последних поисковых запросов, которые пользователь использовал в приложении. Если какой-либо критерий не был задействован в поиске, он не отображается в истории поиска.

Во вкладке «Аналитика» отображаются круговые диаграммы, отражающие статистику поисковых запросов по маркам автомобилей и по состоянию автомобилей. При нажатии на один из элементов легенды диаграмм соответствующий элемент диаграммы временно удаляется и соотношение элементов в выборке пересчитывается. При повторном нажатии на элемент легенды статистика восстанавливается.

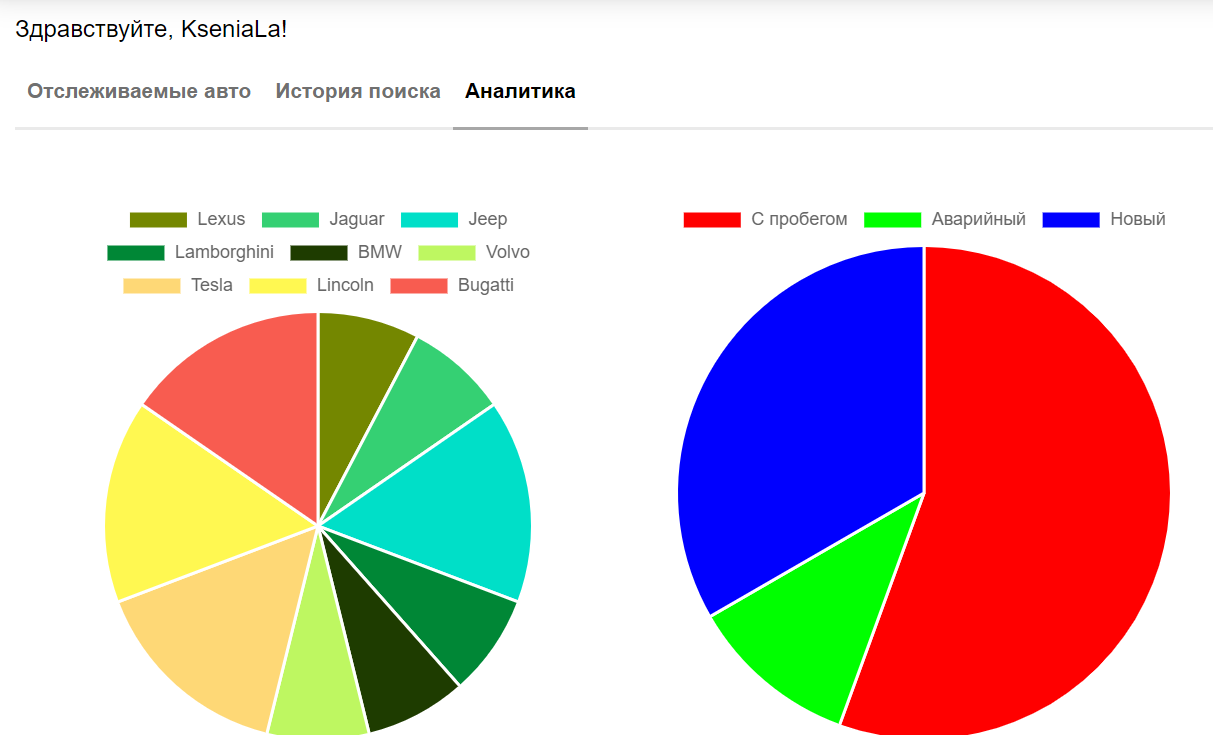


Рисунок 6.3 - Вкладка «Аналитика» в личном кабинете пользователя

**6.2 Указания по установке**

Для нормального функционирования разработанного программного средства на компьютер должны быть установлены следующие программные средства:

* Visual Studio 2017;
* MS SQL Server 2012;
* npm;
* IIS Express.

Сначала необходимо клонировать репозиторий по ссылке <https://github.com/KseniaLa/CarApp.git>. Далее в командной строке нужно перейти в папку «UI» приложения и выполнить команду npm install. После установки всех зависимостей нужно выполнить команду npm run serve и дождаться сборки проекта. Далее веб-приложение доступно по адресу <http://localhost:8081>.

Для запуска сервера необходимо открыть проект в Visual Studio 2017 и выполнить команду Update-Database в командной строке. Затем в Visual Studio нужно нажать кнопку «Пуск». После запуска сервера разработанное приложения полностью готово к работе в режиме разработки.

**ЗАКЛЮЧЕНИЕ**

В ходе выполнения данного курсового проекта было разработано веб-приложение для агрегатора авто-объявлений, имеющее следующие функции: поиск объявлений по различным критериям поиска, добавление объявления в отслеживаемые объявления, отображение последних поисковых запросов пользователя и статистики поисковых запросов. Разработанное программное средство имеет интуитивно понятный интерфейс и рассчитан на работу с большим количеством пользователей.

В ходе проектирования данного веб-приложения были изучены принципы построения одностраничных приложений и REST-архитектуры. Также были закреплены навыки работы с базами данных (на примере MS SQL Server), написания хранимых процедур, функций, представлений. Также были изучены способы взаимодействия серверной части приложения с базой данных.

Итоговая программа была испытана при различных входных данных. Особое внимание было уделено поведению программы в критических ситуациях (прерывание Интернет-соединения, некорректные аутентификационные данные).

Разработанное веб-приложение может быть расширено. Например, может быть добавлено больше критериев поиска, могут быть ускорены алгоритмы поиска объявлений, добавлены уведомления пользователей по электронной почте или в мессенджерах. Также может быть добавлена поддержка большего количества языков приложения.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

[1] Сайты агрегаторы объявлений [Электронный ресурс]. – Электронные данные – Режим доступа: http://www.gaw.ru/aggregators.

[2] Принципы REST [Электронный ресурс]. – Электронные данные – Режим доступа: https://www.codecademy.com/articles/what-is-rest

[3] Haraba.ru [Электронный ресурс]: – Электронные данные – Режим доступа: https://haraba.ru/

[4] Robasta.ru [Электронный ресурс]: – Электронные данные – Режим доступа: https://robasta.ru/

[5] Autobot.by [Электронный ресурс]: – Электронные данные – Режим доступа: <https://autobot.by/>

[6] Vuex Documentation [Электронный ресурс]: – Электронные данные – Режим доступа: https://vuex.vuejs.org/

[7] ASP.NET Core Documentation [Электронный ресурс]: – Электронные данные – Режим доступа: <https://docs.microsoft.com/en-us/aspnet/core/>

[8] Руководство по языку C#. [Электронный ресурс]: – Электронные данные – Режим доступа: https://docs.microsoft.com/ru-ru/dotnet/csharp/

[9] Обзор Microsoft SQL Server 2012 [Электронный ресурс]: – Электронные данные – Режим доступа: <https://www.microsoft.com/ru-ru/sql-server/sql-server-2012>

[10] Обзор технологии ADO.NET [Электронный ресурс]: – Электронные данные – Режим доступа: <http://www.cyberguru.ru/microsoft-net/ado-net>

**ПРИЛОЖЕНИЕ А**

**(обязательное)**

**Текст программного модуля главного графического компонента приложения**

<template>

<div class="center-container">

<div class="center-content">

<input class="modal-input" type="text" v-model="login" placeholder="Логин" />

<input class="modal-input" type="password" v-model="password" placeholder="Пароль" />

<button v-on:click="goToMainPage" class="site-button">Войти</button>

<button v-on:click="goToRegisterPage" class="site-button">Регистрация</button>

</div>

<button v-on:click="guestLogin" class="site-button">Продолжить без авторизации</button>

<modal name="login-error" height="20%" width="40%">

<error-popup v-bind:message="errorMessage" />

</modal>

</div>

</template>

<script>

import router from "../router/index.js";

import ErrorPopup from "../components/ErrorPopup";

import UserFetch from "../dataFetch/userFetch.js";

export default {

name: "LoginPage",

components: {

ErrorPopup

},

data: function() {

return {

login: "",

password: "",

errorMessage: "",

fillMessage: "Заполните все поля",

noUserMessage: "Ошибка авторизациию Проверьте логин и пароль"

};

},

methods: {

guestLogin: function() {

router.push("main");

},

goToMainPage: async function() {

if (!this.login || !this.password) {

this.errorMessage = this.fillMessage;

this.$modal.show("login-error");

return;

}

let user = await UserFetch.getUser(this.login, this.password);

if (user) {

this.$store.commit("setUser", user);

router.push("main");

} else {

this.errorMessage = this.noUserMessage;

this.$modal.show("login-error");

}

},

goToRegisterPage: function() {

router.push("register");

}

}

};

</script>

<style lang="scss">

.sections-container {

display: flex;

flex-direction: row;

height: 100%;

}

.modal-input {

margin-bottom: 10px;

padding: 10px;

border-radius: 5px;

text-decoration: none;

outline: none;

font-size: 20px;

box-shadow: none;

resize: none;

}

.center-container {

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

background: #287728;

height: 100%;

}

.center-content {

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

padding: 60px;

background: #287728;

box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0, 0, 0, 0.19);

@media screen and (max-width: 500px) {

padding: 30px;

}

}

.v--modal-overlay[data-modal="login-error"] {

background: transparent;

}

</style>

**ПРИЛОЖЕНИЕ Б**

**(обязательное)**

**Текст программного модуля контроллера приложения**

using CarApp.Models;

using CarApp.Services.Interfaces;

using Microsoft.AspNetCore.Mvc;

namespace CarApp.Controllers

{

[Route("api/users")]

public class UsersController : Controller

{

private readonly IUserService \_userService;

public UsersController(IUserService userService)

{

\_userService = userService;

}

[HttpGet("login")]

public IActionResult Login(string login, string password)

{

var user = \_userService.GetUser(login, password);

if (user != null)

{

return Ok(new { id = user.UserId, email = user.Email, login = user.Login, isAdmin = user.IsAdmin, IsPro = user.IsPro });

}

return BadRequest();

}

[HttpPost("register")]

public IActionResult Register([FromBody]RegisterModel reg)

{

var userRegistered = \_userService.RegisterUser(reg);

if (userRegistered)

{

return Ok();

}

return BadRequest();

}

[HttpPost("addPro/{id}")]

public IActionResult ApplyForPro(int id)

{

var userIsPro = \_userService.ApplyForPro(id);

if (userIsPro)

{

return Ok();

}

return BadRequest();

}

}

}

**ПРИЛОЖЕНИЕ В**

**(обязательное)**

**Текст программного модуля сервиса приложения**

using CarApp.Models;

using CarApp.Providers;

using CarApp.Providers.Interfaces;

using CarApp.Services.Interfaces;

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Threading.Tasks;

using System.Xml;

using System.Xml.Serialization;

namespace CarApp.Services

{

public class SearchService : ISearchService, IScopedService

{

private readonly ISearchProvider \_searchProvider;

public SearchService()

{

\_searchProvider = new SearchProvider();

}

public BaseSearchParameters GetBaseSearchParameters()

{

return \_searchProvider.GetBaseSearchParameters();

}

public List<KeyValuePair<int, string>> GetCarModelsOptions(int brandId)

{

return \_searchProvider.GetCarModelsOptions(brandId);

}

public List<KeyValuePair<int, string>> GetCarModelsOptions()

{

return \_searchProvider.GetCarModelsOptions();

}

public void SaveUserSearch(Search search, int userId)

{

var searchXml = SerializeSearchToXml(search);

\_searchProvider.SaveUserSearch(userId, searchXml);

}

public List<SearchExtended> GetUserSearchHistory(int userId)

{

var searchHst = \_searchProvider.GetUserSearchHistory(userId);

var searchObjList = DeserializeSearch(searchHst);

var options = GetBaseSearchParameters();

var sites = options.Sites.ToDictionary(x => x.Key);

var bodies = options.CarBodyTypes.ToDictionary(x => x.Key);

var brands = options.CarBrands.ToDictionary(x => x.Key);

var models = GetCarModelsOptions().ToDictionary(x => x.Key);

var conditions = options.CarConditionTypes.ToDictionary(x => x.Key);

var engines = options.CarEngineTypes.ToDictionary(x => x.Key);

var result = new List<SearchExtended>();

foreach(var s in searchObjList)

{

var se = new SearchExtended

{

Brand = GetValueOrNull(brands, s.Brand),

Model = GetValueOrNull(models, s.Model),

BodyType = GetValueOrNull(bodies, s.BodyType),

EngineType = GetValueOrNull(engines, s.EngineType),

ConditionType = GetValueOrNull(conditions, s.ConditionType),

};

se.Sites = new List<KeyValuePair<int, string>>();

if (s.Sites != null)

{

foreach (var site in s.Sites)

{

se.Sites.Add(sites.GetValueOrDefault(site));

}

}

result.Add(se);

}

return result;

}

public Analitics GetAnalitics()

{

var searchHst = \_searchProvider.GetSearchHistory();

var searchObjList = DeserializeSearch(searchHst);

var options = GetBaseSearchParameters();

var brandNames = options.CarBrands.ToDictionary(x => x.Key, v => v.Value);

var conditionNames = options.CarConditionTypes.ToDictionary(x => x.Key, v => v.Value);

var brands = searchObjList.Where(x => x.Brand.HasValue).Select(x => x.Brand.Value);

var conditions = searchObjList.Where(x => x.ConditionType.HasValue).Select(x => x.ConditionType.Value);

var brandCounts = brands.ToLookup(x => x);

var conditionCounts = conditions.ToLookup(x => x);

var brandDict = new Dictionary<int, int>();

var conditionDict = new Dictionary<int, int>();

foreach (var i in brandCounts)

{

brandDict[i.Key] = i.ToList().Count;

}

foreach (var i in conditionCounts)

{

conditionDict[i.Key] = i.ToList().Count;

}

var result = new Analitics();

var brandNameCount = new List<KeyValuePair<string, int>>();

var conditionNameCount = new List<KeyValuePair<string, int>>();

foreach(var b in brandDict)

{

brandNameCount.Add(new KeyValuePair<string, int>(brandNames[b.Key], b.Value));

}

foreach (var c in conditionDict)

{

conditionNameCount.Add(new KeyValuePair<string, int>(conditionNames[c.Key], c.Value));

}

result.BrandNames = brandNameCount.Select(x => x.Key).ToList();

result.BrandCounts = brandNameCount.Select(x => x.Value).ToList();

result.ConditionNames = conditionNameCount.Select(x => x.Key).ToList();

result.ConditionCounts = conditionNameCount.Select(x => x.Value).ToList();

return result;

}

private KeyValuePair<int, string>? GetValueOrNull(Dictionary<int, KeyValuePair<int, string>> dict, int? key)

{

if (!key.HasValue) return null;

if (dict.ContainsKey(key.Value))

{

return dict[key.Value];

}

else

{

return null;

}

}

private string SerializeSearchToXml(Search search)

{

var serializer = new XmlSerializer(typeof(Search));

var value = search;

var xml = "";

using (var sww = new StringWriter())

{

using (var writer = XmlWriter.Create(sww))

{

serializer.Serialize(writer, value);

xml = sww.ToString();

}

}

return xml;

}

private List<Search> DeserializeSearch(List<string> xmlElements)

{

var result = new List<Search>();

var serializer = new XmlSerializer(typeof(Search));

foreach (var xml in xmlElements)

{

using (var reader = new StringReader(xml))

{

result.Add((Search)serializer.Deserialize(reader));

}

}

return result;

}

}

}

**ПРИЛОЖЕНИЕ Г**

**(обязательное)**

**Дамп структуры базы данных**

USE [master]

GO

/\*\*\*\*\*\* Object: Database [FindAndFollow] Script Date: 08.12.2019 15:47:34 \*\*\*\*\*\*/

CREATE DATABASE [FindAndFollow]

CONTAINMENT = NONE

ON PRIMARY

( NAME = N'FindAndFollow', FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL14.SQLEXPRESS\MSSQL\DATA\FindAndFollow\_Primary.mdf' , SIZE = 8192KB , MAXSIZE = UNLIMITED, FILEGROWTH = 65536KB )

LOG ON

( NAME = N'FindAndFollow\_log', FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL14.SQLEXPRESS\MSSQL\DATA\FindAndFollow\_Primary.ldf' , SIZE = 8192KB , MAXSIZE = 2048GB , FILEGROWTH = 65536KB )

GO

ALTER DATABASE [FindAndFollow] SET COMPATIBILITY\_LEVEL = 140

GO

IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))

begin

EXEC [FindAndFollow].[dbo].[sp\_fulltext\_database] @action = 'enable'

end

GO

…

USE [FindAndFollow]

GO

/\*\*\*\*\*\* Object: User [service\_acc] Script Date: 08.12.2019 15:47:34 \*\*\*\*\*\*/

CREATE USER [service\_acc] FOR LOGIN [service\_acc] WITH DEFAULT\_SCHEMA=[dbo]

GO

/\*\*\*\*\*\* Object: UserDefinedFunction [dbo].[DecodeCyrillicJSON] Script Date: 08.12.2019 15:47:34 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE FUNCTION [dbo].[DecodeCyrillicJSON]

(

@pString NVARCHAR(MAX)

)

RETURNS @returntable TABLE

(

String NVARCHAR(MAX)

)

AS

BEGIN

INSERT @returntable

SELECT @pString

RETURN

END

GO

/\*\*\*\*\*\* Object: UserDefinedFunction [dbo].[MappingType] Script Date: 08.12.2019 15:47:34 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE FUNCTION [dbo].[MappingType]

(

@pType VARCHAR(20),

@pString NVARCHAR(100)

)

RETURNS INT

AS

BEGIN

RETURN

CASE

WHEN @pType = 'BodyType' THEN

CASE @pString

WHEN N'Седан' THEN 1

WHEN N'Универсал' THEN 2

WHEN N'Хетчбэк' THEN 3

WHEN N'Минивэн' THEN 4

WHEN N'Внедорожник' THEN 5

WHEN N'Купе' THEN 6

WHEN N'Кабриолет' THEN 7

WHEN N'Микроавтобус' THEN 8

WHEN N'Грузовик' THEN 9

WHEN N'Пикап' THEN 10

WHEN N'Родстер' THEN 11

WHEN N'Автобус' THEN 12

ELSE 0

END

WHEN @pType = 'EngineType' THEN

CASE @pString

WHEN N'Бензиновый' THEN 1

WHEN N'Дизельный' THEN 2

WHEN N'Газ' THEN 3

WHEN N'Гибридный бензиновый' THEN 4

WHEN N'Гибридный дизельный' THEN 5

WHEN N'Электрический' THEN 6

ELSE 0

END

WHEN @pType = 'TransmissionType' THEN

CASE @pString

WHEN N'Автомат' THEN 1

WHEN N'Механика' THEN 2

ELSE 0

END

WHEN @pType = 'DriveType' THEN

CASE @pString

WHEN N'Передний' THEN 1

WHEN N'Задний' THEN 2

WHEN N'Полный' THEN 3

ELSE 0

END

WHEN @pType = 'Condition' THEN

CASE @pString

WHEN N'Новый' THEN 1

WHEN N'С пробегом' THEN 2

WHEN N'Аварийный' THEN 3

ELSE 0

END

WHEN @pType = 'SellerType' THEN

CASE @pString

WHEN N'Частное' THEN 1

WHEN N'Автохаус' THEN 2

WHEN N'Дилер' THEN 3

ELSE 0

END

ELSE 0

END

END

GO

/\*\*\*\*\*\* Object: UserDefinedFunction [dbo].[ParseJSON] Script Date: 08.12.2019 15:47:34 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE FUNCTION [dbo].[ParseJSON]( @JSON NVARCHAR(MAX))

RETURNS @hierarchy TABLE

(

Element\_ID INT IDENTITY(1, 1) NOT NULL, /\* internal surrogate primary key gives the order of parsing and the list order \*/

SequenceNo [int] NULL, /\* the place in the sequence for the element \*/

Parent\_ID INT,/\* if the element has a parent then it is in this column. The document is the ultimate parent, so you can get the structure from recursing from the document \*/

Object\_ID INT,/\* each list or object has an object id. This ties all elements to a parent. Lists are treated as objects here \*/

Name NVARCHAR(2000),/\* the Name of the object \*/

StringValue NVARCHAR(MAX) NOT NULL,/\*the string representation of the value of the element. \*/

ValueType NVARCHAR(10) NOT NULL /\* the declared type of the value represented as a string in StringValue\*/

)

AS

BEGIN

DECLARE

@FirstObject INT, --the index of the first open bracket found in the JSON string

@OpenDelimiter INT,--the index of the next open bracket found in the JSON string

@NextOpenDelimiter INT,--the index of subsequent open bracket found in the JSON string

@NextCloseDelimiter INT,--the index of subsequent close bracket found in the JSON string

@Type NVARCHAR(10),--whether it denotes an object or an array

@NextCloseDelimiterChar CHAR(1),--either a '}' or a ']'

@Contents NVARCHAR(MAX), --the unparsed contents of the bracketed expression

@Start INT, --index of the start of the token that you are parsing

@end INT,--index of the end of the token that you are parsing

@param INT,--the parameter at the end of the next Object/Array token

@EndOfName INT,--the index of the start of the parameter at end of Object/Array token

@token NVARCHAR(200),--either a string or object

@value NVARCHAR(MAX), -- the value as a string

@SequenceNo int, -- the sequence number within a list

@Name NVARCHAR(200), --the Name as a string

@Parent\_ID INT,--the next parent ID to allocate

@lenJSON INT,--the current length of the JSON String

@characters NCHAR(63),--used to convert hex to decimal

@result BIGINT,--the value of the hex symbol being parsed

@index SMALLINT,--used for parsing the hex value

@Escape INT --the index of the next escape character

…

/\*\*\*\*\*\* Object: Table [dbo].[Car] Script Date: 08.12.2019 15:47:34 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Car](

[CarId] [int] IDENTITY(1,1) NOT NULL,

[CarModelId] [int] NOT NULL,

[SiteId] [int] NOT NULL,

[PlaceId] [int] NOT NULL,

[Price] [money] NOT NULL,

[BodyType] [int] NOT NULL,

[ModelYear] [smallint] NOT NULL,

[EngineType] [int] NOT NULL,

[EngineSize] [smallint] NOT NULL,

[TransmissionType] [int] NOT NULL,

[DriveType] [int] NULL,

[Condition] [int] NOT NULL,

[Mileage] [int] NULL,

[ColorId] [tinyint] NOT NULL,

[SellerType] [int] NOT NULL,

[IsSwap] [bit] NOT NULL,

[Description] [nvarchar](4000) NULL,

[OriginalURL] [nvarchar](100) NULL,

[PageCreatedOn] [datetime2](7) NULL,

[CarImages] [nvarchar](4000) NULL,

[OptionList] [nvarchar](4000) NULL,

[OriginalCarBrand] [nvarchar](50) NULL,

[OriginalCarModel] [nvarchar](50) NULL,

[IsActive] [bit] NOT NULL,

[CreatedOn] [datetime2](7) NOT NULL,

[CurrencyId] [int] NULL,

CONSTRAINT [pkCar] PRIMARY KEY CLUSTERED

(

[CarId] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: View [dbo].[vwCarActive] Script Date: 08.12.2019 15:47:34 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE VIEW [dbo].[vwCarActive]

AS

SELECT CarId,

OriginalURL

FROM dbo.Car

WHERE IsActive = 1

GO

/\*\*\*\*\*\* Object: Table [dbo].[CarBrand] Script Date: 08.12.2019 15:47:34 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[CarBrand](

[CarBrandId] [int] IDENTITY(1,1) NOT NULL,

[Name] [nvarchar](50) NOT NULL,

[CreatedOn] [datetime2](7) NOT NULL,

CONSTRAINT [pkCarBrand] PRIMARY KEY CLUSTERED

(

[CarBrandId] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[CarModel] Script Date: 08.12.2019 15:47:34 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[CarModel](

[CarModelId] [int] IDENTITY(1,1) NOT NULL,

[CarBrandId] [int] NOT NULL,

[Name] [nvarchar](50) NOT NULL,

[CreatedOn] [datetime2](7) NOT NULL,

CONSTRAINT [pkCarModel] PRIMARY KEY CLUSTERED

(

[CarModelId] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: View [dbo].[vwCarBrandModel] Script Date: 08.12.2019 15:47:34 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE VIEW [dbo].[vwCarBrandModel]

AS

SELECT cb.CarBrandId,

cb.Name AS CarBrandName,

cm.CarModelId,

cm.Name AS CarModelName

FROM dbo.[CarBrand] cb

JOIN dbo.[CarModel] cm ON cm.CarBrandId = cb.CarBrandId

/\*

SELECT \*

FROM vwCarBrandModel

WHERE CarBrandName = N''

\*/

GO

/\*\*\*\*\*\* Object: Table [dbo].[CarOwnerPhone] Script Date: 08.12.2019 15:47:34 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[CarOwnerPhone](

[CarOwnerPhoneId] [int] IDENTITY(1,1) NOT NULL,

[CarId] [int] NOT NULL,

[Phone] [varchar](30) NOT NULL,

[IsActual] [bit] NOT NULL,

[CreatedOn] [datetime2](7) NOT NULL,

CONSTRAINT [pkCarOwnerPhone] PRIMARY KEY CLUSTERED

(

[CarOwnerPhoneId] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[City] Script Date: 08.12.2019 15:47:34 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[City](

[CityId] [int] IDENTITY(1,1) NOT NULL,

[Name] [nvarchar](100) NOT NULL,

[CreatedOn] [datetime2](7) NOT NULL,

CONSTRAINT [pkCity] PRIMARY KEY CLUSTERED

(

[CityId] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Color] Script Date: 08.12.2019 15:47:34 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Color](

[ColorId] [tinyint] IDENTITY(1,1) NOT NULL,

[Name] [nvarchar](100) NOT NULL,

[CreatedOn] [datetime2](7) NOT NULL,

CONSTRAINT [pkColor] PRIMARY KEY CLUSTERED

(

[ColorId] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Country] Script Date: 08.12.2019 15:47:34 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Country](

[CountryId] [int] IDENTITY(1,1) NOT NULL,

[Name] [nvarchar](100) NOT NULL,

[CreatedOn] [datetime2](7) NOT NULL,

CONSTRAINT [pkCountry] PRIMARY KEY CLUSTERED

(

[CountryId] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[CurrencyRate] Script Date: 08.12.2019 15:47:34 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[CurrencyRate](

[CurrencyId] [int] NOT NULL,

[RateDate] [date] NOT NULL,

[Rate] [money] NOT NULL,

CONSTRAINT [pkCurrencyRate] PRIMARY KEY CLUSTERED

(

[CurrencyId] ASC,

[RateDate] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Place] Script Date: 08.12.2019 15:47:34 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Place](

[PlaceId] [int] IDENTITY(1,1) NOT NULL,

[CountryId] [int] NOT NULL,

[RegionId] [int] NOT NULL,

[CityId] [int] NOT NULL,

[CreatedOn] [datetime2](7) NOT NULL,

CONSTRAINT [pkPlace] PRIMARY KEY CLUSTERED

(

[PlaceId] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Region] Script Date: 08.12.2019 15:47:34 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Region](

[RegionId] [int] IDENTITY(1,1) NOT NULL,

[Name] [nvarchar](100) NOT NULL,

[CreatedOn] [datetime2](7) NOT NULL,

CONSTRAINT [pkRegion] PRIMARY KEY CLUSTERED

(

[RegionId] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Site] Script Date: 08.12.2019 15:47:34 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Site](

[SiteId] [int] IDENTITY(1,1) NOT NULL,

[SiteUrl] [nvarchar](100) NOT NULL,

[ShortDescription] [nvarchar](100) NOT NULL,

[FullDescription] [nvarchar](500) NOT NULL,

[CountryId] [int] NOT NULL,

[CreatedOn] [datetime2](7) NOT NULL,

CONSTRAINT [pkSite] PRIMARY KEY CLUSTERED

(

[SiteId] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: View [dbo].[vwCar] Script Date: 08.12.2019 15:47:34 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE VIEW [dbo].[vwCar]

AS

SELECT

c.CarId

,cb.Name AS CarBrand

,c.OriginalCarBrand

,cm.Name AS CarModel

,c.OriginalCarModel

,s.SiteUrl

,c2.Name AS Country

,r.Name AS Region

,c3.Name AS City

,c.Price

,c.Price / cr.Rate AS USDPrice

,CASE c.BodyType

WHEN 1 THEN 'Sedan'

WHEN 2 THEN 'Wagon'

WHEN 3 THEN 'Hatchback'

WHEN 4 THEN 'Minivan'

WHEN 5 THEN 'SUV'

WHEN 6 THEN 'Coupe'

WHEN 7 THEN 'Cabriolet'

WHEN 8 THEN 'Minibus'

WHEN 9 THEN 'Van'

WHEN 10 THEN 'Pickup'

WHEN 11 THEN 'Roadster'

WHEN 12 THEN 'Bus'

WHEN 0 THEN 'Other'

ELSE NULL

END AS BodyType

,c.ModelYear

,CASE c.EngineType

WHEN 1 THEN 'Petrol'

WHEN 2 THEN 'Diesel'

WHEN 3 THEN 'Gas'

WHEN 4 THEN 'Hybrid (petrol)'

WHEN 5 THEN 'Hybrid (diesel)'

WHEN 6 THEN 'Electric'

WHEN 0 THEN 'Other'

ELSE NULL

END AS EngineType

,c.EngineSize

,CASE c.TransmissionType

WHEN 1 THEN 'Automatic'

WHEN 2 THEN 'Manual'

WHEN 0 THEN 'Other'

ELSE NULL

END AS TransmissionType

,CASE c.DriveType

WHEN 1 THEN 'FWD'

WHEN 2 THEN 'RWD'

WHEN 3 THEN '4x4'

WHEN 0 THEN 'Other'

ELSE NULL

END AS DriveType

,CASE c.Condition

WHEN 1 THEN 'New'

WHEN 2 THEN 'Used'

WHEN 3 THEN 'Wreck'

WHEN 0 THEN 'Other'

ELSE NULL

END AS Condition

,c.Mileage

,c4.Name AS Color

,CASE c.SellerType

WHEN 1 THEN 'Private'

WHEN 2 THEN 'Autohaus'

WHEN 3 THEN 'Dieler'

WHEN 0 THEN 'Other'

ELSE NULL

END AS SellerType

,cop.Phone1

,cop.Phone2

,cop.Phone3

,c.IsSwap

,c.Description

,c.OriginalURL

,c.PageCreatedOn

,f.StringValue AS CarImages

,c.OptionList

,c.CreatedOn

FROM dbo.Car c

JOIN dbo.CarModel AS cm ON cm.CarModelId = c.CarModelId

JOIN dbo.CarBrand AS cb ON cb.CarBrandId = cm.CarBrandId

JOIN dbo.[Site] AS s ON s.SiteId = c.SiteId

JOIN dbo.Place AS p ON p.PlaceId = c.PlaceId

JOIN dbo.Country AS c2 ON c2.CountryId = p.CountryId

JOIN dbo.Region AS r ON r.RegionId = p.RegionId

JOIN dbo.City AS c3 ON c3.CityId = p.CityId

JOIN dbo.Color AS c4 ON c4.ColorId = c.ColorId

LEFT JOIN

( SELECT CarId, Phone1, Phone2, Phone3

FROM (SELECT CarId, 'Phone'+CAST(ROW\_NUMBER() OVER (PARTITION BY CarId ORDER BY CarId DESC) AS VARCHAR) AS PhoneId, Phone FROM dbo.CarOwnerPhone) x

PIVOT (MAX(Phone) FOR PhoneId IN ([Phone1], [Phone2], [Phone3]) ) pvt

) AS cop ON cop.CarId = c.CarId

LEFT JOIN dbo.CurrencyRate AS cr ON cr.CurrencyId = 145 AND cr.RateDate = CAST(GETDATE() AS DATE)

CROSS APPLY dbo.ParseJSON(c.CarImages) f

WHERE c.IsActive = 1

AND f.NAME = 'ImageURL'

AND f.Element\_ID = 1

GO

/\*\*\*\*\*\* Object: View [dbo].[vwCarDuplicateCar] Script Date: 08.12.2019 15:47:34 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE VIEW [dbo].[vwCarDuplicateCar]

AS

SELECT CarBrand,

CarModel,

Country,

Region,

City,

BodyType,

ModelYear,

EngineType,

EngineSize,

TransmissionType,

DriveType,

Condition,

Color,

SellerType,

Mileage,

LEFT(Description,100) AS Description100,

COUNT(\*) AS CarCount

FROM dbo.vwCar

GROUP BY CarBrand,

CarModel,

Country,

Region,

City,

BodyType,

ModelYear,

EngineType,

EngineSize,

TransmissionType,

DriveType,

Condition,

Color,

SellerType,

Mileage,

LEFT(Description,100)

HAVING COUNT(\*) > 1

GO

/\*\*\*\*\*\* Object: View [dbo].[vwCarDuplicateDescription] Script Date: 08.12.2019 15:47:34 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE VIEW [dbo].[vwCarDuplicateDescription]

AS

SELECT LEFT(Description,100) AS Description,

Phone1 AS Phone,

COUNT(\*) AS CarCount

FROM dbo.vwCar

WHERE ISNULL(LEFT(Description,100),'') <> ''

GROUP BY LEFT(Description,100),Phone1

HAVING COUNT(\*) > 1

GO

/\*\*\*\*\*\* Object: View [dbo].[vwCarDuplicatePhone] Script Date: 08.12.2019 15:47:34 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE VIEW [dbo].[vwCarDuplicatePhone]

AS

SELECT Phone1 AS Phone,

COUNT(\*) AS CarCount

FROM dbo.vwCar

WHERE Phone1 IS NOT NULL

AND SellerType <> 'Autohaus'

GROUP BY Phone1

HAVING COUNT(\*) > 1

GO

/\*\*\*\*\*\* Object: View [dbo].[vwCarInfo] Script Date: 08.12.2019 15:47:34 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE VIEW [dbo].[vwCarInfo]

AS

SELECT

c.CarId

,cb.Name AS CarBrand

,cm.Name AS CarModel

,s.SiteUrl

,c2.Name AS Country

,r.Name AS Region

,c3.Name AS City

,c.Price

,CASE c.BodyType

WHEN 1 THEN 'Sedan'

WHEN 2 THEN 'Wagon'

WHEN 3 THEN 'Hatchback'

WHEN 4 THEN 'Minivan'

WHEN 5 THEN 'SUV'

WHEN 6 THEN 'Coupe'

WHEN 7 THEN 'Cabriolet'

WHEN 8 THEN 'Minibus'

WHEN 9 THEN 'Van'

WHEN 10 THEN 'Pickup'

WHEN 11 THEN 'Roadster'

WHEN 12 THEN 'Bus'

WHEN 0 THEN 'Other'

ELSE NULL

END AS BodyType

,c.ModelYear

,CASE c.EngineType

WHEN 1 THEN 'Petrol'

WHEN 2 THEN 'Diesel'

WHEN 3 THEN 'Gas'

WHEN 4 THEN 'Hybrid (petrol)'

WHEN 5 THEN 'Hybrid (diesel)'

WHEN 6 THEN 'Electric'

WHEN 0 THEN 'Other'

ELSE NULL

END AS EngineType

,c.EngineSize

,CASE c.TransmissionType

WHEN 1 THEN 'Automatic'

WHEN 2 THEN 'Manual'

WHEN 0 THEN 'Other'

ELSE NULL

END AS TransmissionType

,CASE c.DriveType

WHEN 1 THEN 'FWD'

WHEN 2 THEN 'RWD'

WHEN 3 THEN '4x4'

WHEN 0 THEN 'Other'

ELSE NULL

END AS DriveType

,CASE c.Condition

WHEN 1 THEN 'New'

WHEN 2 THEN 'Used'

WHEN 3 THEN 'Wreck'

WHEN 0 THEN 'Other'

ELSE NULL

END AS Condition

,c.Mileage

,c4.Name AS Color

,CASE c.SellerType

WHEN 1 THEN 'Private'

WHEN 2 THEN 'Autohaus'

WHEN 3 THEN 'Dieler'

WHEN 0 THEN 'Other'

ELSE NULL

END AS SellerType

,c.IsSwap

,c.Description

,c.OriginalURL

,c.PageCreatedOn

,c.CreatedOn

FROM dbo.Car c

JOIN dbo.CarModel AS cm ON cm.CarModelId = c.CarModelId

JOIN dbo.CarBrand AS cb ON cb.CarBrandId = cm.CarBrandId

JOIN dbo.[Site] AS s ON s.SiteId = c.SiteId

JOIN dbo.Place AS p ON p.PlaceId = c.PlaceId

JOIN dbo.Country AS c2 ON c2.CountryId = p.CountryId

JOIN dbo.Region AS r ON r.RegionId = p.RegionId

JOIN dbo.City AS c3 ON c3.CityId = p.CityId

JOIN dbo.Color AS c4 ON c4.ColorId = c.ColorId

WHERE c.IsActive = 1;

GO

/\*\*\*\*\*\* Object: Table [dbo].[BodyType] Script Date: 08.12.2019 15:47:34 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[BodyType](

[ID] [int] IDENTITY(1,1) NOT NULL,

[Name] [nvarchar](50) NOT NULL,

CONSTRAINT [PK\_BodyType] PRIMARY KEY CLUSTERED

(

[ID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[CarBrandException] Script Date: 08.12.2019 15:47:34 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[CarBrandException](

[CarBrandExceptionId] [int] IDENTITY(1,1) NOT NULL,

[ExceptionMask] [nvarchar](100) NOT NULL,

CONSTRAINT [pkCarBrandException] PRIMARY KEY CLUSTERED

(

[CarBrandExceptionId] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[CarBrandMapping] Script Date: 08.12.2019 15:47:34 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[CarBrandMapping](

[CarBrandMappingId] [int] IDENTITY(1,1) NOT NULL,

[BrandMask] [nvarchar](100) NOT NULL,

[CarBrandId] [int] NOT NULL,

CONSTRAINT [pkCarBrandMapping] PRIMARY KEY CLUSTERED

(

[CarBrandMappingId] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[CarLog] Script Date: 08.12.2019 15:47:34 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[CarLog](

[CarLogId] [int] IDENTITY(1,1) NOT NULL,

[CarId] [int] NULL,

[Price] [money] NULL,

[Mileage] [int] NULL,

[SellerType] [tinyint] NULL,

[Description] [nvarchar](4000) NULL,

[PageCreatedOn] [datetime2](7) NULL,

[CreatedOn] [datetime2](7) NULL,

CONSTRAINT [pkCarLog] PRIMARY KEY CLUSTERED

(

[CarLogId] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[CarModelMapping] Script Date: 08.12.2019 15:47:35 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[CarModelMapping](

[CarModelMappingId] [int] IDENTITY(1,1) NOT NULL,

[ModelMask] [nvarchar](100) NOT NULL,

[ModelNotMask] [nvarchar](100) NOT NULL,

[CarBrandId] [int] NOT NULL,

[CarModelId] [int] NOT NULL,

CONSTRAINT [pkCarModelMapping] PRIMARY KEY CLUSTERED

(

[CarModelMappingId] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[CarParsing] Script Date: 08.12.2019 15:47:35 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

…

/\*\*\*\*\*\* Object: Table [dbo].[Currency] Script Date: 08.12.2019 15:47:35 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Currency](

[CurrencyId] [int] NOT NULL,

[CharCode] [varchar](3) NOT NULL,

[EnglishName] [varchar](255) NOT NULL,

[CreatedOn] [datetime2](7) NOT NULL,

CONSTRAINT [pkCurrency] PRIMARY KEY CLUSTERED

(

[CurrencyId] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[DriveType] Script Date: 08.12.2019 15:47:35 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[DriveType](

[ID] [int] IDENTITY(1,1) NOT NULL,

[Name] [nvarchar](50) NOT NULL,

CONSTRAINT [PK\_DriveType] PRIMARY KEY CLUSTERED

(

[ID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Log] Script Date: 08.12.2019 15:47:35 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Log](

[LogId] [int] IDENTITY(1,1) NOT NULL,

[UserId] [int] NOT NULL,

[LogMessage] [nvarchar](4000) NOT NULL,

[CreatedOn] [datetime2](7) NOT NULL,

CONSTRAINT [pkLog] PRIMARY KEY CLUSTERED

(

[LogId] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Search] Script Date: 08.12.2019 15:47:35 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Search](

[SearchId] [int] IDENTITY(1,1) NOT NULL,

[UserId] [int] NOT NULL,

[SearchXML] [xml] NOT NULL,

[IsActive] [bit] NOT NULL,

[CreatedOn] [datetime2](7) NOT NULL,

[ModifiedOn] [datetime2](7) NULL,

CONSTRAINT [pkSearch] PRIMARY KEY CLUSTERED

(

[SearchId] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[SearchSite] Script Date: 08.12.2019 15:47:35 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[SearchSite](

[SearchSiteId] [int] IDENTITY(1,1) NOT NULL,

[SearchId] [int] NOT NULL,

[SiteId] [int] NOT NULL,

[CreatedOn] [datetime2](7) NOT NULL,

CONSTRAINT [pkSearchSite] PRIMARY KEY CLUSTERED

(

[SearchSiteId] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[SellerType] Script Date: 08.12.2019 15:47:35 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[SellerType](

[ID] [int] IDENTITY(1,1) NOT NULL,

[Name] [nvarchar](50) NOT NULL,

CONSTRAINT [PK\_SellerType] PRIMARY KEY CLUSTERED

(

[ID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Subscription] Script Date: 08.12.2019 15:47:35 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Subscription](

[SubscriptionId] [int] IDENTITY(1,1) NOT NULL,

[UserId] [int] NOT NULL,

[DateStart] [date] NOT NULL,

[DateEnd] [date] NOT NULL,

[Type] [tinyint] NOT NULL,

[CreatedOn] [datetime2](7) NOT NULL,

CONSTRAINT [pkSubscription] PRIMARY KEY CLUSTERED

(

[SubscriptionId] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[TransmissionType] Script Date: 08.12.2019 15:47:35 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[TransmissionType](

[ID] [int] IDENTITY(1,1) NOT NULL,

[Name] [nvarchar](50) NOT NULL,

CONSTRAINT [PK\_TransmissionType] PRIMARY KEY CLUSTERED

(

[ID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[User] Script Date: 08.12.2019 15:47:35 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[User](

[UserId] [int] IDENTITY(1,1) NOT NULL,

[Login] [varchar](30) NOT NULL,

[Password] [varchar](10) NOT NULL,

[Email] [varchar](30) NOT NULL,

[IsActive] [bit] NOT NULL,

[CreatedOn] [datetime2](7) NOT NULL,

[ModifiedOn] [datetime2](7) NULL,

CONSTRAINT [pkUser] PRIMARY KEY CLUSTERED

(

[UserId] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[UserCarSubscription] Script Date: 08.12.2019 15:47:35 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[UserCarSubscription](

[ID] [int] IDENTITY(1,1) NOT NULL,

[UserID] [int] NOT NULL,

[CarID] [int] NOT NULL,

CONSTRAINT [PK\_UserCarSubscription] PRIMARY KEY CLUSTERED

(

[ID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

SET ANSI\_PADDING ON

GO

/\*\*\*\*\*\* Object: Index [ixCar\_OriginalURL\_inc] Script Date: 08.12.2019 15:47:35 \*\*\*\*\*\*/

CREATE UNIQUE NONCLUSTERED INDEX [ixCar\_OriginalURL\_inc] ON [dbo].[Car]

(

[OriginalURL] ASC

)

INCLUDE ( [IsActive]) WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, SORT\_IN\_TEMPDB = OFF, IGNORE\_DUP\_KEY = OFF, DROP\_EXISTING = OFF, ONLINE = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

GO

SET ANSI\_PADDING ON

GO

/\*\*\*\*\*\* Object: Index [ixCarBrand\_Name] Script Date: 08.12.2019 15:47:35 \*\*\*\*\*\*/

CREATE UNIQUE NONCLUSTERED INDEX [ixCarBrand\_Name] ON [dbo].[CarBrand]

(

[Name] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, SORT\_IN\_TEMPDB = OFF, IGNORE\_DUP\_KEY = OFF, DROP\_EXISTING = OFF, ONLINE = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

GO

SET ANSI\_PADDING ON

GO

/\*\*\*\*\*\* Object: Index [ixCarModel\_CarBrandId\_Name] Script Date: 08.12.2019 15:47:35 \*\*\*\*\*\*/

CREATE UNIQUE NONCLUSTERED INDEX [ixCarModel\_CarBrandId\_Name] ON [dbo].[CarModel]

(

[CarBrandId] ASC,

[Name] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, SORT\_IN\_TEMPDB = OFF, IGNORE\_DUP\_KEY = OFF, DROP\_EXISTING = OFF, ONLINE = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

GO

SET ANSI\_PADDING ON

GO

/\*\*\*\*\*\* Object: Index [ixCarOwnerPhone\_CarId\_Phone] Script Date: 08.12.2019 15:47:35 \*\*\*\*\*\*/

CREATE UNIQUE NONCLUSTERED INDEX [ixCarOwnerPhone\_CarId\_Phone] ON [dbo].[CarOwnerPhone]

(

[CarId] ASC,

[Phone] ASC,

[IsActual] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, SORT\_IN\_TEMPDB = OFF, IGNORE\_DUP\_KEY = OFF, DROP\_EXISTING = OFF, ONLINE = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

GO

SET ANSI\_PADDING ON

GO

/\*\*\*\*\*\* Object: Index [ixCity\_Name] Script Date: 08.12.2019 15:47:35 \*\*\*\*\*\*/

CREATE UNIQUE NONCLUSTERED INDEX [ixCity\_Name] ON [dbo].[City]

(

[Name] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, SORT\_IN\_TEMPDB = OFF, IGNORE\_DUP\_KEY = OFF, DROP\_EXISTING = OFF, ONLINE = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

GO

SET ANSI\_PADDING ON

GO

/\*\*\*\*\*\* Object: Index [ixColor\_Name] Script Date: 08.12.2019 15:47:35 \*\*\*\*\*\*/

CREATE UNIQUE NONCLUSTERED INDEX [ixColor\_Name] ON [dbo].[Color]

(

[Name] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, SORT\_IN\_TEMPDB = OFF, IGNORE\_DUP\_KEY = OFF, DROP\_EXISTING = OFF, ONLINE = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

GO

SET ANSI\_PADDING ON

GO

/\*\*\*\*\*\* Object: Index [ixCurrency\_CharCode] Script Date: 08.12.2019 15:47:35 \*\*\*\*\*\*/

CREATE NONCLUSTERED INDEX [ixCurrency\_CharCode] ON [dbo].[Currency]

(

[CharCode] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, SORT\_IN\_TEMPDB = OFF, DROP\_EXISTING = OFF, ONLINE = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Index [ixPlace\_CityId\_CountryId\_RegionId] Script Date: 08.12.2019 15:47:35 \*\*\*\*\*\*/

CREATE UNIQUE NONCLUSTERED INDEX [ixPlace\_CityId\_CountryId\_RegionId] ON [dbo].[Place]

(

[CityId] ASC,

[CountryId] ASC,

[RegionId] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, SORT\_IN\_TEMPDB = OFF, IGNORE\_DUP\_KEY = OFF, DROP\_EXISTING = OFF, ONLINE = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

GO

SET ANSI\_PADDING ON

GO

/\*\*\*\*\*\* Object: Index [ixSite\_SiteURL\_CountryId] Script Date: 08.12.2019 15:47:35 \*\*\*\*\*\*/

CREATE UNIQUE NONCLUSTERED INDEX [ixSite\_SiteURL\_CountryId] ON [dbo].[Site]

(

[SiteUrl] ASC,

[CountryId] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, SORT\_IN\_TEMPDB = OFF, IGNORE\_DUP\_KEY = OFF, DROP\_EXISTING = OFF, ONLINE = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Car] ADD CONSTRAINT [dfCar\_IsActive] DEFAULT ((1)) FOR [IsActive]

GO

ALTER TABLE [dbo].[Car] ADD CONSTRAINT [dfCar\_CreatedOn] DEFAULT (getutcdate()) FOR [CreatedOn]

GO

ALTER TABLE [dbo].[CarBrand] ADD CONSTRAINT [dfCarBrand\_CreatedOn] DEFAULT (getutcdate()) FOR [CreatedOn]

GO

ALTER TABLE [dbo].[CarLog] ADD CONSTRAINT [dfCarLog\_CreatedOn] DEFAULT (getutcdate()) FOR [CreatedOn]

GO

ALTER TABLE [dbo].[CarModel] ADD CONSTRAINT [dfCarModel\_CreatedOn] DEFAULT (getutcdate()) FOR [CreatedOn]

GO

ALTER TABLE [dbo].[CarOwnerPhone] ADD CONSTRAINT [dfCarOwnerPhone\_IsActual] DEFAULT ((1)) FOR [IsActual]

GO

ALTER TABLE [dbo].[CarOwnerPhone] ADD CONSTRAINT [dfCarOwnerPhone\_CreatedOn] DEFAULT (getutcdate()) FOR [CreatedOn]

GO

ALTER TABLE [dbo].[CarParsing] ADD CONSTRAINT [dfCarParsing\_CreatedOn] DEFAULT (getutcdate()) FOR [CreatedOn]

GO

ALTER TABLE [dbo].[CarParsing] ADD CONSTRAINT [dfCarParsing\_PageStatusId] DEFAULT ((1)) FOR [PageStatusId]

GO

ALTER TABLE [dbo].[City] ADD CONSTRAINT [dfCity\_CreatedOn] DEFAULT (getutcdate()) FOR [CreatedOn]

GO

ALTER TABLE [dbo].[Color] ADD CONSTRAINT [dfColor\_CreatedOn] DEFAULT (getutcdate()) FOR [CreatedOn]

GO

ALTER TABLE [dbo].[Country] ADD CONSTRAINT [dfCountry\_CreatedOn] DEFAULT (getutcdate()) FOR [CreatedOn]

GO

ALTER TABLE [dbo].[Currency] ADD CONSTRAINT [dfCurrency\_CreatedOn] DEFAULT (getutcdate()) FOR [CreatedOn]

GO

ALTER TABLE [dbo].[ErrorLog] ADD CONSTRAINT [dfErrorLog\_CreatedOn] DEFAULT (getutcdate()) FOR [CreatedOn]

GO

ALTER TABLE [dbo].[Log] ADD CONSTRAINT [dfLog\_CreatedOn] DEFAULT (getutcdate()) FOR [CreatedOn]

GO

ALTER TABLE [dbo].[Place] ADD CONSTRAINT [dfPlace\_CreatedOn] DEFAULT (getutcdate()) FOR [CreatedOn]

GO

ALTER TABLE [dbo].[Region] ADD CONSTRAINT [dfRegion\_CreatedOn] DEFAULT (getutcdate()) FOR [CreatedOn]

GO

ALTER TABLE [dbo].[Search] ADD CONSTRAINT [dfSearch\_CreatedOn] DEFAULT (getutcdate()) FOR [CreatedOn]

GO

ALTER TABLE [dbo].[SearchSite] ADD CONSTRAINT [dfSearchSite\_CreatedOn] DEFAULT (getutcdate()) FOR [CreatedOn]

GO

ALTER TABLE [dbo].[Site] ADD CONSTRAINT [dfSite\_CreatedOn] DEFAULT (getutcdate()) FOR [CreatedOn]

GO

ALTER TABLE [dbo].[Subscription] ADD CONSTRAINT [dfSubscription\_CreatedOn] DEFAULT (getutcdate()) FOR [CreatedOn]

GO

ALTER TABLE [dbo].[User] ADD CONSTRAINT [dfUser\_CreatedOn] DEFAULT (getutcdate()) FOR [CreatedOn]

GO

ALTER TABLE [dbo].[Car] WITH CHECK ADD CONSTRAINT [FK\_Car\_BodyType] FOREIGN KEY([BodyType])

REFERENCES [dbo].[BodyType] ([ID])

GO

ALTER TABLE [dbo].[Car] CHECK CONSTRAINT [FK\_Car\_BodyType]

GO

ALTER TABLE [dbo].[Car] WITH CHECK ADD CONSTRAINT [FK\_Car\_Condition] FOREIGN KEY([Condition])

REFERENCES [dbo].[Condition] ([ID])

GO

ALTER TABLE [dbo].[Car] CHECK CONSTRAINT [FK\_Car\_Condition]

GO

ALTER TABLE [dbo].[Car] WITH CHECK ADD CONSTRAINT [FK\_Car\_Currency] FOREIGN KEY([CurrencyId])

REFERENCES [dbo].[Currency] ([CurrencyId])

GO

ALTER TABLE [dbo].[Car] CHECK CONSTRAINT [FK\_Car\_Currency]

GO

ALTER TABLE [dbo].[Car] WITH CHECK ADD CONSTRAINT [FK\_Car\_DriveType] FOREIGN KEY([DriveType])

REFERENCES [dbo].[DriveType] ([ID])

GO

ALTER TABLE [dbo].[Car] CHECK CONSTRAINT [FK\_Car\_DriveType]

GO

ALTER TABLE [dbo].[Car] WITH CHECK ADD CONSTRAINT [FK\_Car\_EngineType] FOREIGN KEY([EngineType])

REFERENCES [dbo].[EngineType] ([ID])

GO

ALTER TABLE [dbo].[Car] CHECK CONSTRAINT [FK\_Car\_EngineType]

GO

ALTER TABLE [dbo].[Car] WITH CHECK ADD CONSTRAINT [FK\_Car\_SellerType] FOREIGN KEY([SellerType])

REFERENCES [dbo].[SellerType] ([ID])

GO

ALTER TABLE [dbo].[Car] CHECK CONSTRAINT [FK\_Car\_SellerType]

GO

ALTER TABLE [dbo].[Car] WITH CHECK ADD CONSTRAINT [FK\_Car\_TransmissionType] FOREIGN KEY([TransmissionType])

REFERENCES [dbo].[TransmissionType] ([ID])

GO

ALTER TABLE [dbo].[Car] CHECK CONSTRAINT [FK\_Car\_TransmissionType]

GO

ALTER TABLE [dbo].[Car] WITH CHECK ADD CONSTRAINT [fkCar\_CarModel] FOREIGN KEY([CarModelId])

REFERENCES [dbo].[CarModel] ([CarModelId])

GO

ALTER TABLE [dbo].[Car] CHECK CONSTRAINT [fkCar\_CarModel]

GO

ALTER TABLE [dbo].[Car] WITH CHECK ADD CONSTRAINT [fkCar\_Color] FOREIGN KEY([ColorId])

REFERENCES [dbo].[Color] ([ColorId])

GO

ALTER TABLE [dbo].[Car] CHECK CONSTRAINT [fkCar\_Color]

GO

ALTER TABLE [dbo].[Car] WITH CHECK ADD CONSTRAINT [fkCar\_Place] FOREIGN KEY([PlaceId])

REFERENCES [dbo].[Place] ([PlaceId])

GO

ALTER TABLE [dbo].[Car] CHECK CONSTRAINT [fkCar\_Place]

GO

ALTER TABLE [dbo].[Car] WITH CHECK ADD CONSTRAINT [fkCar\_Site] FOREIGN KEY([SiteId])

REFERENCES [dbo].[Site] ([SiteId])

GO

ALTER TABLE [dbo].[Car] CHECK CONSTRAINT [fkCar\_Site]

GO

ALTER TABLE [dbo].[CarBrandMapping] WITH CHECK ADD CONSTRAINT [fkCarBrandMapping\_CarBrand] FOREIGN KEY([CarBrandId])

REFERENCES [dbo].[CarBrand] ([CarBrandId])

GO

ALTER TABLE [dbo].[CarBrandMapping] CHECK CONSTRAINT [fkCarBrandMapping\_CarBrand]

GO

ALTER TABLE [dbo].[CarLog] WITH CHECK ADD CONSTRAINT [fkCarLog\_Car] FOREIGN KEY([CarId])

REFERENCES [dbo].[Car] ([CarId])

GO

ALTER TABLE [dbo].[CarLog] CHECK CONSTRAINT [fkCarLog\_Car]

GO

ALTER TABLE [dbo].[CarModel] WITH CHECK ADD CONSTRAINT [fkCarModel\_CarBrand] FOREIGN KEY([CarBrandId])

REFERENCES [dbo].[CarBrand] ([CarBrandId])

GO

ALTER TABLE [dbo].[CarModel] CHECK CONSTRAINT [fkCarModel\_CarBrand]

GO

ALTER TABLE [dbo].[CarModelMapping] WITH CHECK ADD CONSTRAINT [fkCarModelMapping\_CarBrand] FOREIGN KEY([CarBrandId])

REFERENCES [dbo].[CarBrand] ([CarBrandId])

GO

ALTER TABLE [dbo].[CarModelMapping] CHECK CONSTRAINT [fkCarModelMapping\_CarBrand]

GO

ALTER TABLE [dbo].[CarModelMapping] WITH CHECK ADD CONSTRAINT [fkCarModelMapping\_CarModel] FOREIGN KEY([CarModelId])

REFERENCES [dbo].[CarModel] ([CarModelId])

GO

ALTER TABLE [dbo].[CarModelMapping] CHECK CONSTRAINT [fkCarModelMapping\_CarModel]

GO

ALTER TABLE [dbo].[CarOwnerPhone] WITH CHECK ADD CONSTRAINT [fkCarOwnerPhone\_Car] FOREIGN KEY([CarId])

REFERENCES [dbo].[Car] ([CarId])

GO

ALTER TABLE [dbo].[CarOwnerPhone] CHECK CONSTRAINT [fkCarOwnerPhone\_Car]

GO

ALTER TABLE [dbo].[CurrencyRate] WITH CHECK ADD CONSTRAINT [fkCurrencyRate\_Currency] FOREIGN KEY([CurrencyId])

REFERENCES [dbo].[Currency] ([CurrencyId])

GO

ALTER TABLE [dbo].[CurrencyRate] CHECK CONSTRAINT [fkCurrencyRate\_Currency]

GO

ALTER TABLE [dbo].[Log] WITH CHECK ADD CONSTRAINT [fkLog\_User] FOREIGN KEY([UserId])

REFERENCES [dbo].[User] ([UserId])

GO

ALTER TABLE [dbo].[Log] CHECK CONSTRAINT [fkLog\_User]

GO

ALTER TABLE [dbo].[Place] WITH CHECK ADD CONSTRAINT [fkPlace\_City] FOREIGN KEY([CityId])

REFERENCES [dbo].[City] ([CityId])

GO

ALTER TABLE [dbo].[Place] CHECK CONSTRAINT [fkPlace\_City]

GO

ALTER TABLE [dbo].[Place] WITH CHECK ADD CONSTRAINT [fkPlace\_Country] FOREIGN KEY([CountryId])

REFERENCES [dbo].[Country] ([CountryId])

GO

ALTER TABLE [dbo].[Place] CHECK CONSTRAINT [fkPlace\_Country]

GO

ALTER TABLE [dbo].[Place] WITH CHECK ADD CONSTRAINT [fkPlace\_Region] FOREIGN KEY([RegionId])

REFERENCES [dbo].[Region] ([RegionId])

GO

ALTER TABLE [dbo].[Place] CHECK CONSTRAINT [fkPlace\_Region]

GO

ALTER TABLE [dbo].[Search] WITH CHECK ADD CONSTRAINT [fkSearch\_User] FOREIGN KEY([UserId])

REFERENCES [dbo].[User] ([UserId])

GO

ALTER TABLE [dbo].[Search] CHECK CONSTRAINT [fkSearch\_User]

GO

ALTER TABLE [dbo].[SearchSite] WITH CHECK ADD CONSTRAINT [fkSearchSite\_Search] FOREIGN KEY([SearchId])

REFERENCES [dbo].[Search] ([SearchId])

GO

ALTER TABLE [dbo].[SearchSite] CHECK CONSTRAINT [fkSearchSite\_Search]

GO

ALTER TABLE [dbo].[SearchSite] WITH CHECK ADD CONSTRAINT [fkSearchSite\_Site] FOREIGN KEY([SiteId])

REFERENCES [dbo].[Site] ([SiteId])

GO

ALTER TABLE [dbo].[SearchSite] CHECK CONSTRAINT [fkSearchSite\_Site]

GO

ALTER TABLE [dbo].[Site] WITH CHECK ADD CONSTRAINT [fkSite\_Country] FOREIGN KEY([CountryId])

REFERENCES [dbo].[Country] ([CountryId])

GO

ALTER TABLE [dbo].[Site] CHECK CONSTRAINT [fkSite\_Country]

GO

ALTER TABLE [dbo].[Subscription] WITH CHECK ADD CONSTRAINT [fkSubscription\_User] FOREIGN KEY([UserId])

REFERENCES [dbo].[User] ([UserId])

GO

ALTER TABLE [dbo].[Subscription] CHECK CONSTRAINT [fkSubscription\_User]

GO

ALTER TABLE [dbo].[UserCarSubscription] WITH CHECK ADD CONSTRAINT [FK\_UserCarSubscription\_Car] FOREIGN KEY([CarID])

REFERENCES [dbo].[Car] ([CarId])

GO

ALTER TABLE [dbo].[UserCarSubscription] CHECK CONSTRAINT [FK\_UserCarSubscription\_Car]

GO

ALTER TABLE [dbo].[UserCarSubscription] WITH CHECK ADD CONSTRAINT [FK\_UserCarSubscription\_User] FOREIGN KEY([UserID])

REFERENCES [dbo].[User] ([UserId])

GO

ALTER TABLE [dbo].[UserCarSubscription] CHECK CONSTRAINT [FK\_UserCarSubscription\_User]

GO

ALTER TABLE [dbo].[Car] WITH CHECK ADD CONSTRAINT [ckCar\_BodyType] CHECK (([BodyType]>=(0) AND [BodyType]<=(12)))

GO

ALTER TABLE [dbo].[Car] CHECK CONSTRAINT [ckCar\_BodyType]

GO

ALTER TABLE [dbo].[Car] WITH CHECK ADD CONSTRAINT [ckCar\_Condition] CHECK (([Condition]>=(0) AND [Condition]<=(3)))

GO

ALTER TABLE [dbo].[Car] CHECK CONSTRAINT [ckCar\_Condition]

GO

ALTER TABLE [dbo].[Car] WITH CHECK ADD CONSTRAINT [ckCar\_DriveType] CHECK (([DriveType]>=(0) AND [DriveType]<=(3)))

GO

ALTER TABLE [dbo].[Car] CHECK CONSTRAINT [ckCar\_DriveType]

GO

ALTER TABLE [dbo].[Car] WITH CHECK ADD CONSTRAINT [ckCar\_EngineType] CHECK (([EngineType]>=(0) AND [EngineType]<=(6)))

GO

ALTER TABLE [dbo].[Car] CHECK CONSTRAINT [ckCar\_EngineType]

GO

ALTER TABLE [dbo].[Car] WITH CHECK ADD CONSTRAINT [ckCar\_SellerType] CHECK (([SellerType]>=(0) AND [SellerType]<=(3)))

GO

ALTER TABLE [dbo].[Car] CHECK CONSTRAINT [ckCar\_SellerType]

GO

ALTER TABLE [dbo].[Car] WITH CHECK ADD CONSTRAINT [ckCar\_TransmissionType] CHECK (([TransmissionType]>=(0) AND [TransmissionType]<=(2)))

GO

ALTER TABLE [dbo].[Car] CHECK CONSTRAINT [ckCar\_TransmissionType]

GO

ALTER TABLE [dbo].[Subscription] WITH CHECK ADD CONSTRAINT [ckSubscription\_Type] CHECK (([Type]>=(0) AND [Type]<=(5)))

GO

ALTER TABLE [dbo].[Subscription] CHECK CONSTRAINT [ckSubscription\_Type]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[CarLogInsert] Script Date: 08.12.2019 15:47:35 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROCEDURE [dbo].[CarLogInsert]

AS

BEGIN

SET NOCOUNT ON;

BEGIN TRY

BEGIN TRANSACTION

INSERT INTO dbo.CarLog

(

[CarId],

[Price],

[Mileage],

[SellerType],

[Description],

[PageCreatedOn]

)

SELECT

[CarId],

[Price],

[Mileage],

[SellerType],

[Description],

[PageCreatedOn]

FROM #Car

;

COMMIT TRANSACTION

END TRY

BEGIN CATCH

IF @@TRANCOUNT > 0

BEGIN

ROLLBACK TRANSACTION;

END

EXECUTE dbo.ErrorLogInsert @pSendEmail = 1;

END CATCH

END

/\*

exec dbo.CarLogInsert

\*/

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[CarOwnerPhoneInsert] Script Date: 08.12.2019 15:47:35 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROCEDURE [dbo].[CarOwnerPhoneInsert]

AS

BEGIN

BEGIN TRY

BEGIN TRANSACTION

SET NOCOUNT ON;

UPDATE cop

SET cop.IsActual = 0

FROM dbo.CarOwnerPhone AS cop

JOIN #Car AS t ON t.CarId = cop.CarId

WHERE cop.IsActual = 1

;

INSERT INTO [dbo].[CarOwnerPhone]

(

[CarId],

[Phone]

)

SELECT DISTINCT

t.CarId,

f.Stringvalue

FROM #Car t

CROSS APPLY dbo.ParseJSON (t.OwnerPhone) f

WHERE f.Name = 'phoneNumber'

AND t.OwnerPhone IS NOT NULL

;

COMMIT TRANSACTION

END TRY

BEGIN CATCH

IF @@TRANCOUNT > 0

BEGIN

ROLLBACK TRANSACTION;

END

EXECUTE dbo.ErrorLogInsert @pSendEmail = 1;

END CATCH

END

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[CarParsingInsert] Script Date: 08.12.2019 15:47:35 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROCEDURE [dbo].[CarParsingInsert]

(

@pCarBrand NVARCHAR(50),

@pModel NVARCHAR(50),

@pSiteId NVARCHAR(50),

@pSiteURL NVARCHAR(1000),

@pCity NVARCHAR(100),

@pCountry NVARCHAR(100),

@pPrice NVARCHAR(100),

@pBodyType NVARCHAR(100),

@pModelYear NVARCHAR(4),

@pEngineType NVARCHAR(100),

@pEngineSize NVARCHAR(5),

@pTransmissionType NVARCHAR(100),

@pDriveType NVARCHAR(100),

@pCondition NVARCHAR(100),

@pMileage NVARCHAR(100),

@pColor NVARCHAR(100),

@pSellerType NVARCHAR(100),

@pIsSwap CHAR(1),

@pDescription NVARCHAR(4000),

@pOwnerPhone NVARCHAR(300),

@pCarImages NVARCHAR(4000),

@pOptionList NVARCHAR(4000),

@pPageCreatedOn NVARCHAR(100)

)

AS

BEGIN

SET NOCOUNT ON;

BEGIN TRY

BEGIN TRANSACTION

INSERT INTO dbo.CarParsing

(

CarBrand,

Model,

SiteId,

SiteUrl,

City,

Country,

Price,

BodyType,

ModelYear,

EngineType,

EngineSize,

TransmissionType,

DriveType,

Condition,

Mileage,

Color,

SellerType,

IsSwap,

[Description],

OwnerPhone,

CarImages,

OptionList,

PageCreatedOn

)

SELECT

@pCarBrand,

@pModel,

@pSiteId,

@pSiteURL,

@pCity,

@pCountry,

@pPrice,

@pBodyType,

@pModelYear,

@pEngineType,

@pEngineSize,

@pTransmissionType,

@pDriveType,

@pCondition,

@pMileage,

@pColor,

@pSellerType,

@pIsSwap,

@pDescription,

@pOwnerPhone,

@pCarImages,

@pOptionList,

@pPageCreatedOn

;

COMMIT TRANSACTION

END TRY

BEGIN CATCH

IF @@TRANCOUNT > 0

BEGIN

ROLLBACK TRANSACTION;

END

EXECUTE dbo.ErrorLogInsert @pSendEmail = 1;

END CATCH

END

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[CarParsingMergeCar] Script Date: 08.12.2019 15:47:35 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROCEDURE [dbo].[CarParsingMergeCar]

(

@pDebug BIT = 0

)

AS

BEGIN

SET NOCOUNT ON

;

IF OBJECT\_ID('tempdb..#CarParsing') IS NOT NULL DROP TABLE #CarParsing

;

IF OBJECT\_ID('tempdb..#Car') IS NOT NULL DROP TABLE #Car

;

CREATE TABLE #Car

(

CarId INT,

OwnerPhone NVARCHAR(300),

OriginalURL NVARCHAR(1000),

Price MONEY,

Mileage INT,

SellerType TINYINT,

Description NVARCHAR(4000),

PageCreatedOn DATETIME2

)

;

DECLARE @XML XML,

@ErrorNumber INT = 0,

@ErrorSeverity INT = 1,

@ErrorState INT = 16,

@ErrorObject NVARCHAR(126) = OBJECT\_NAME(@@PROCID),

@ErrorMessageShort NVARCHAR(1000),

@ErrorMessageFull NVARCHAR(MAX)

;

BEGIN TRY

BEGIN TRANSACTION

-- load and parse data

SELECT

CASE

WHEN cbe.CarBrandExceptionId IS NOT NULL THEN 'Exception Error'

WHEN COALESCE(cb.CarBrandId,cbm.CarBrandId) IS NULL THEN 'CarBrand Error'

WHEN COALESCE(cm.CarModelId,cmm.CarModelId) IS NULL THEN 'CarModel Error'

WHEN s.SiteId IS NULL THEN 'Site Error'

WHEN c.CityId IS NULL THEN 'City Error'

WHEN cn.CountryId IS NULL THEN 'Country Error'

WHEN pl.PlaceId IS NULL THEN 'Place Error'

WHEN col.ColorId IS NULL THEN 'Color Error'

WHEN cp.CarBrand IS NULL THEN 'Original CarBrand Error'

WHEN cp.Model IS NULL THEN 'Original CarModel Error'

WHEN cp.ModelYear IS NULL THEN 'Original ModelYear Error'

WHEN cp.Color IS NULL THEN 'Original Color Error'

WHEN cp.EngineType IS NULL THEN 'Original EngineType Error'

WHEN cp.EngineSize IS NULL THEN 'Original EngineSize Error'

WHEN cp.BodyType IS NULL THEN 'Original BodyType Error'

--WHEN cp.Mileage IS NULL THEN 'Original Mileage Error'

WHEN cp.Condition IS NULL THEN 'Original Condition Error'

WHEN cp.TransmissionType IS NULL THEN 'Original TransmissionType Error'

--WHEN cp.DriveType IS NULL THEN 'Original DriveType Error'

WHEN cp.Price IS NULL THEN 'Original Price Error'

--WHEN cp.OwnerPhone IS NULL THEN 'Original OwnerPhone Error'

WHEN cp.City IS NULL THEN 'Original City Error'

WHEN cp.Country IS NULL THEN 'Original Country Error'

ELSE NULL

END AS ErrorType,

cp.CarParsingId,

COALESCE(cb.CarBrandId,cbm.CarBrandId) AS CarBrandId,

COALESCE(cb.Name,cb2.Name) AS CarBrand,

cp.CarBrand AS OriginalCarBrand,

COALESCE(cm.CarModelId,cmm.CarModelId) AS CarModelId,

COALESCE(cm.Name,cm2.Name) AS Model,

cp.Model AS OriginalCarModel,

s.SiteId,

cp.SiteId AS SiteURL,

c.CityId,

cp.City,

cn.CountryId,

cp.Country,

pl.PlaceId,

cp.Price,

cp.BodyType,

cp.ModelYear,

cp.EngineType,

cp.EngineSize,

cp.TransmissionType,

cp.DriveType,

cp.Condition,

cp.Mileage,

col.ColorId,

cp.Color,

cp.SellerType,

cp.IsSwap,

cp.Description,

cp.OwnerPhone,

cp.CarImages,

cp.OptionList,

cp.OriginalURL,

cp.PageCreatedOn

INTO #CarParsing

FROM (

SELECT cp.CarParsingId,

cp.CarBrand,

cp.Model,

cp.SiteId,

CASE WHEN cp.City = N'Могилёв' THEN N'Могилев' ELSE cp.City END AS City,

cp.Country,

cp.Price,

dbo.MappingType('BodyType',cp.BodyType) AS BodyType,

cp.ModelYear,

dbo.MappingType('EngineType',cp.EngineType) AS EngineType,

cp.EngineSize,

dbo.MappingType('TransmissionType',cp.TransmissionType) AS TransmissionType,

dbo.MappingType('DriveType',cp.DriveType) AS DriveType,

dbo.MappingType('Condition',cp.Condition) AS Condition,

cp.Mileage,

cp.Color,

dbo.MappingType('SellerType',cp.SellerType) AS SellerType,

cp.IsSwap,

cp.Description,

cp.OwnerPhone,

cp.CarImages,

cp.OptionList,

cp.SiteUrl AS OriginalURL,

cp.PageCreatedOn

FROM dbo.CarParsing cp

WHERE cp.PageStatusId = 1 -- Downloaded page (default)

) AS cp

LEFT JOIN dbo.CarBrand AS cb ON cb.Name = cp.CarBrand

LEFT JOIN dbo.CarBrandMapping AS cbm

ON cb.CarBrandId IS NULL

AND cp.CarBrand LIKE cbm.BrandMask

LEFT JOIN dbo.CarBrand AS cb2 ON cb2.CarBrandId = cbm.CarBrandId

LEFT JOIN dbo.CarModel AS cm ON cm.Name = cp.Model AND cm.CarBrandId = COALESCE(cb.CarBrandId,cb2.CarBrandId)

LEFT JOIN dbo.[Site] AS s ON s.SiteUrl = cp.SiteId

LEFT JOIN dbo.Color AS col ON col.Name = cp.Color

LEFT JOIN dbo.City AS c ON c.Name = cp.City

LEFT JOIN dbo.Country AS cn ON cn.Name = cp.Country

LEFT JOIN dbo.Place AS pl ON pl.CityId = c.CityId AND pl.CountryId = cn.CountryId

LEFT JOIN dbo.CarModelMapping AS cmm

ON cm.CarModelId IS NULL

AND cp.Model LIKE cmm.ModelMask

AND cp.Model NOT LIKE cmm.ModelNotMask

AND cmm.CarBrandId = cb.CarBrandId

LEFT JOIN dbo.CarModel AS cm2 ON cm2.CarModelId = cmm.CarModelId

LEFT JOIN dbo.CarBrandException AS cbe ON cp.CarBrand LIKE cbe.ExceptionMask

;

-- write wrong data info to Log table

IF EXISTS (SELECT 1 FROM #CarParsing WHERE ErrorType IS NOT NULL) BEGIN

SELECT @XML = (

SELECT cp.ErrorType,

cp.CarParsingId,

cp.CarBrand,

cp.OriginalCarBrand,

cp.Model,

cp.OriginalCarModel,

cp.SiteId,

cp.City,

cp.Country,

cp.Price,

cp.BodyType,

cp.ModelYear,

cp.EngineType,

cp.EngineSize,

cp.TransmissionType,

cp.DriveType,

cp.Condition,

cp.Mileage,

cp.Color,

cp.SellerType,

cp.IsSwap,

cp.Description,

cp.OwnerPhone,

cp.CarImages,

cp.OptionList,

cp.OriginalURL,

cp.PageCreatedOn

FROM #CarParsing AS cp

WHERE cp.ErrorType IS NOT NULL

FOR XML PATH, ROOT

)

;

SELECT @ErrorMessageShort = 'Wrong data in CarParsing table',

@ErrorMessageFull = dbo.XMLToJSON(@XML)

;

EXECUTE dbo.ErrorLogInsert

@pErrorNumber = @ErrorNumber,

@pErrorSeverity = @ErrorSeverity,

@pErrorState = @ErrorState,

@pErrorObject = @ErrorObject,

@pErrorMessageShort = @ErrorMessageShort,

@pErrorMessageFull = @ErrorMessageFull,

@pSendEmail = 1

;

IF @pDebug = 1 BEGIN

PRINT 'Merge errors:' + CHAR(10) + @ErrorMessageFull;

END

;

END

;

-- merge with Car table

MERGE dbo.Car AS target

USING

(

SELECT cp.CarModelId,

cp.SiteId,

cp.PlaceId,

cp.Price,

cp.BodyType,

cp.ModelYear,

cp.EngineType,

cp.EngineSize,

cp.TransmissionType,

cp.DriveType,

cp.Condition,

cp.Mileage,

cp.ColorId,

cp.SellerType,

cp.IsSwap,

cp.Description,

cp.OriginalURL,

cp.PageCreatedOn,

cp.CarImages,

cp.OptionList,

cp.OriginalCarBrand,

cp.OriginalCarModel

FROM #CarParsing AS cp

WHERE cp.ErrorType IS NULL

) AS source

ON target.OriginalURL = source.OriginalURL

WHEN MATCHED AND target.PageCreatedOn <> source.PageCreatedOn

THEN

UPDATE

SET

CarModelId = source.CarModelId,

SiteId = source.SiteId,

PlaceId = source.PlaceId,

Price = source.Price,

BodyType = source.BodyType,

ModelYear = source.ModelYear,

EngineType = source.EngineType,

EngineSize = source.EngineSize,

TransmissionType = source.TransmissionType,

DriveType = source.DriveType,

Condition = source.Condition,

Mileage = source.Mileage,

ColorId = source.ColorId,

SellerType = source.SellerType,

IsSwap = source.IsSwap,

Description = source.Description,

OriginalURL = source.OriginalURL,

PageCreatedOn = source.PageCreatedOn,

CarImages = source.CarImages,

OptionList = source.OptionList,

OriginalCarBrand = source.OriginalCarBrand,

OriginalCarModel = source.OriginalCarModel

WHEN NOT MATCHED BY TARGET

THEN

INSERT

(

CarModelId,

SiteId,

PlaceId,

Price,

BodyType,

ModelYear,

EngineType,

EngineSize,

TransmissionType,

DriveType,

Condition,

Mileage,

ColorId,

SellerType,

IsSwap,

Description,

OriginalURL,

PageCreatedOn,

CarImages,

OptionList,

OriginalCarBrand,

OriginalCarModel

)

VALUES

(

source.CarModelId,

source.SiteId,

source.PlaceId,

source.Price,

source.BodyType,

source.ModelYear,

source.EngineType,

source.EngineSize,

source.TransmissionType,

source.DriveType,

source.Condition,

source.Mileage,

source.ColorId,

source.SellerType,

source.IsSwap,

source.Description,

source.OriginalURL,

source.PageCreatedOn,

source.CarImages,

source.OptionList,

source.OriginalCarBrand,

source.OriginalCarModel

)

--when not matched by source then delete

--OUTPUT $action, deleted.\*, inserted.\*

OUTPUT inserted.CarId,

NULL,

inserted.OriginalURL,

inserted.Price,

inserted.Mileage,

inserted.SellerType,

inserted.Description,

inserted.PageCreatedOn

INTO #Car

;

--UPDATE cop

--SET cop.OwnerPhone = cp.OwnerPhone

--FROM #Car cop

--JOIN CarParsing cp ON cp.SiteUrl = cop.OriginalURL

--;

--EXEC dbo.CarOwnerPhoneInsert

--;

UPDATE cp

SET PageStatusId = CASE WHEN t.ErrorType IS NULL THEN 2 ELSE 0 END

FROM dbo.CarParsing cp

JOIN #CarParsing t ON t.CarParsingId = cp.CarParsingId

;

EXEC dbo.CarLogInsert

;

DELETE FROM dbo.CarParsing WHERE PageStatusId <> 1;

;

COMMIT TRANSACTION

END TRY

BEGIN CATCH

IF @@TRANCOUNT > 0

BEGIN

ROLLBACK TRANSACTION;

END

EXECUTE dbo.ErrorLogInsert @pSendEmail = 1;

END CATCH

IF OBJECT\_ID('tempdb..#CarParsing') IS NOT NULL DROP TABLE #CarParsing

;

IF OBJECT\_ID('tempdb..#Car') IS NOT NULL DROP TABLE #Car

;

END

/\*

exec [dbo].[CarParsingMergeCar]

\*/

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[CarParsingSettingsGet] Script Date: 08.12.2019 15:47:35 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

…

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[CarUpdate\_IsActive] Script Date: 08.12.2019 15:47:35 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROCEDURE [dbo].[CarUpdate\_IsActive]

@pCarId INT,

@pIsActive BIT = 0

AS

BEGIN

BEGIN TRY

BEGIN TRANSACTION

SET NOCOUNT ON;

UPDATE dbo.Car

SET IsActive = @pIsActive

WHERE CarId = @pCarId

;

COMMIT TRANSACTION

END TRY

BEGIN CATCH

IF @@TRANCOUNT > 0

BEGIN

ROLLBACK TRANSACTION;

END

EXECUTE dbo.ErrorLogInsert @pSendEmail = 1;

END CATCH

END

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[CurrencyMerge] Script Date: 08.12.2019 15:47:35 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROCEDURE [dbo].[CurrencyMerge]

(

@pDebug bit = 0

)

AS

BEGIN

set nocount on;

DECLARE @ErrCode Int

,@Handle Int

,@URL SysName

,@xml nvarchar(max)

;

SELECT @URL = 'http://www.nbrb.by/Services/XmlExRatesRef.aspx';

EXEC @ErrCode = dbo.HTTPCall @URL, @XML out IF (@ErrCode != 0) RETURN @@Error;

EXEC @ErrCode = sys.sp\_xml\_preparedocument @Handle OUT, @XML IF (@ErrCode != 0) BEGIN RAISERROR('Error parsing XML',18,1) RETURN @@Error END;

MERGE [dbo].[Currency] AS target

USING

(

SELECT

0 as CurrencyId

,'BYN' as CharCode

,'Belorussian Ruble' as EnglishName

UNION ALL

SELECT

CurrencyId

,CharCode

,EnglishName

FROM

(

SELECT

CurrencyId

,CharCode

,EnglishName

FROM OpenXML(@Handle,'//DailyExRates/Currency')

WITH

(

CurrencyId int './@Id',

CharCode varchar(3) './CharCode',

EnglishName varchar(255) './EnglishName'

)

) AS t

WHERE CharCode IN ('USD', 'EUR', 'RUB')

) as source

on target.CurrencyId = source.CurrencyId

when matched

then update set

CharCode = source.CharCode

,EnglishName = source.EnglishName

when not matched by target then insert

(

CurrencyId

,CharCode

,EnglishName

)

values

(

source.CurrencyId

,source.CharCode

,source.EnglishName

)

when not matched by source then delete;

if @pDebug = 1 select @xml;

EXEC @ErrCode = sys.sp\_xml\_removedocument @Handle;

END

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[CurrencyRateGet] Script Date: 08.12.2019 15:47:35 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROCEDURE [dbo].[CurrencyRateGet]

@pCurrencyCode varchar(3) = 'USD'

,@pFromDate date = NULL

,@pToDate date = NULL

,@pDebug bit = 0

AS

BEGIN

SET NOCOUNT ON;

DECLARE @ErrCode Int

,@Handle Int

,@URL SysName

,@xml nvarchar(max)

,@ErrorMessage varchar(2000)

,@CurrencyId int

,@RowCnt int

;

if DATEDIFF(day,@pFromDate,@pToDate) > 365 raiserror('Time interval should be less than one year.',18,1);

if coalesce(@pToDate,@pFromDate) is null begin

select @pFromDate = current\_timestamp

,@pToDate = current\_timestamp

end;

select @CurrencyId = max(CurrencyId)

from dbo.Currency

where CharCode = @pCurrencyCode;

if @CurrencyId is null begin

raiserror ('Entered CurrencyKey did not found. Please use correct CurrencyKey.',18,1);

return;

end;

select @url = 'http://www.nbrb.by/Services/XmlExRatesDyn.aspx?curId=' + cast(@CurrencyId as varchar) + '&fromDate=' + cast(@pFromDate as varchar) + '&toDate=' + cast(@pToDate as varchar) + '';

EXEC @ErrCode = dbo.HTTPCall @URL, @XML out IF (@ErrCode != 0) RETURN @@Error;

EXEC @ErrCode = sys.sp\_xml\_preparedocument @Handle OUT, @XML IF (@ErrCode != 0) BEGIN RAISERROR('Error parsing XML',18,1) RETURN @@Error END;

if @pDebug = 1 select @XML;

exec sp\_xml\_preparedocument @Handle output, @XML;

MERGE [dbo].[CurrencyRate] AS target

USING

(

SELECT @CurrencyId as CurrencyId,

RateDate,

Rate

FROM OpenXML(@Handle,'//Currency/Record')

WITH (

RateDate date './@Date',

Rate money './Rate'

)

) as source

on target.CurrencyId = source.CurrencyId and target.RateDate = source.RateDate

when matched

then update set

Rate = source.Rate

when not matched by target then insert

(

CurrencyId

,RateDate

,Rate

)

values (

source.CurrencyId

,source.RateDate

,source.Rate

)

--when not matched by source then delete

;

exec sp\_xml\_removedocument @Handle;

END

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[ErrorInfoGet] Script Date: 08.12.2019 15:47:35 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROCEDURE [dbo].[ErrorInfoGet]

AS

BEGIN

SET NOCOUNT ON;

PRINT 'Error ' + CONVERT(varchar(50), ERROR\_NUMBER()) +

', Severity ' + CONVERT(varchar(5), ERROR\_SEVERITY()) +

', State ' + CONVERT(varchar(5), ERROR\_STATE()) +

', Procedure ' + ISNULL(ERROR\_PROCEDURE(), '-') +

', Line ' + CONVERT(varchar(5), ERROR\_LINE());

PRINT ERROR\_MESSAGE();

END

GO

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Обозначение | | | | Наименование | | | | Дополнительные сведения | | | |
|  | | | | Текстовые документы | | | |  | | | |
|  | | | |  | | | |  | | | |
| БГУИР КП 1–40 01 01 014 ПЗ | | | | Пояснительная записка | | | | 65 с. | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | | Графические документы | | | |  | | | |
|  | | | |  | | | |  | | | |
| БГУИР 651001  014 СД | | | | Схема данных | | | | Формат А1 | | | |
| БГУИР 651001  014 МБ | | | | Модель базы данных | | | | Формат А1 | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  |  |  |  |  | БГУИР КП 1-40 01 01 014 Д1 | | | | | | |
|  |  |  |  |  |
|  |  |  |  |  |
| Изм. | Л. | № докум. | Подп. | Дата | Веб-приложение для  агрегатора объявлений о продаже автомбилей  Ведомость курсового  проекта |  | | | | Лист | Листов |
| Разраб. | | Лащиновская К.Д. |  |  | Т |  | |  | 65 | 65 |
| Пров. | | Фадеева Е.Е. |  |  | Кафедра ПОИТ  гр. 651001 | | | | | |
|  | |  |  |  |