Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ОРЕНБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет математики и информационных технологий

Кафедра геометрии и компьютерных наук

**КУРСОВАЯ РАБОТА**

по дисциплине «Объектно-ориентированные языки и системы»

**Разработка системы информационной поддержки службы такси**

ОГУ 02.03.02. 3022. 337 OO

Руководитель

Старший преподаватель

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_А.А. Горелик

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2022 г.

Студент группы 20ФИИТ(б)ОП

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_А.Е. Поляков

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2022 г.

Оренбург 2022

Утверждаю

заведующий кафедрой геометрии и компьютерных наук

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_А.Е. Шухман

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2022 г.

**Задание на выполнение курсовой работы**

студенту Полякову Александру Евгеньевичу

по направлению подготовки 02.03.02 Фундаментальная информатика и информационные технологии

1 Тема работы: Разработка системы информационной поддержки службы такси.

2 Срок сдачи студентом работы: «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2022 г.

3 Цель и задачи работы:

Цель – создание WEB-приложения для удобного и быстрого заказа такси с ПК.

Задачи:

1. посмотреть и изучить аналоги;
2. сформулировать бизнес-логику разрабатываемого продукта;
3. разработать схему и модели базы данных, которые будут подходить для данной предметной области;
4. разработать WEB-приложение;
5. протестировать полученное WEB-приложение.

4 Исходные данные к работе: интернет-источники.

5 Перечень вопросов, подлежащих разработке: описание процесса разработки WEB-приложения.

6 Перечень графического (иллюстративного) материала: рисунки и таблица, необходимые для изложения материала.

Дата выдачи и получения задания

Руководитель «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2022 г. А.А. Горелик

Студент «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2022 г. А.Е. Поляков

**Аннотация**

Данная курсовая работа посвящена разработке WEB-приложении по заказу такси. В курсовой работе были проанализированы, выделены достоинства и недостатки уже существующих приложений для заказа такси. Изучив приложения, были поставлены требования для данной разработки. В данной работе большая часть уделена программному коду, а именно описанию классов, методов, взаимодействия между ними и описания что за что отвечает. Так же спроектирована база данных, а именно структура, схема базы данных, связи между таблицами таблицы.

Работа содержит 18 страниц, 4 рисунка, 1 таблицу.

Содержание

[Введение 5](#_Toc131080128)

[1 Анализ рынка приложений по заказу такси 6](#_Toc131080129)

[2 Проектирование базы данных 8](#_Toc131080130)

[3 Разработка программного приложения 9](#_Toc131080131)

[3.1 Создание моделей 9](#_Toc131080132)

[3.2 Создание базы данных 11](#_Toc131080133)

[3.3 Создание контроллеров 12](#_Toc131080134)

[3.4 Создание представлений 14](#_Toc131080135)

[4 Тестирование созданного программного приложения 15](#_Toc131080136)

[Заключение 17](#_Toc131080137)

[Список использованных источников 18](#_Toc131080138)

# Введение

WEB**-приложение** – программа с определенным набором функционала, использующая в качестве клиента браузер. Другими словами, если приложению для осуществления бизнес-логики требуется сетевое соединение и наличие на стороне пользователя браузера, то его относят к WEB-приложению.

Целью курсовой работы является создание WEB-приложения для удобного и быстрого заказа такси с ПК.

Для достижения этой цели выделены следующие задачи:

1. посмотреть и изучить аналоги;
2. сформулировать бизнес-логику разрабатываемого продукта;
3. разработать схему и модели базы данных, которые будут подходить для данной предметной области;
4. разработать WEB-приложение;
5. протестировать полученное WEB-приложение.

# 1 Анализ рынка приложений по заказу такси

Первым для рассмотрения и анализа уже существующего WEB-приложения было выбрано приложение «Яндекс Go». На рисунке 1 представлен внешний вид WEB-приложения «Яндекс Go».

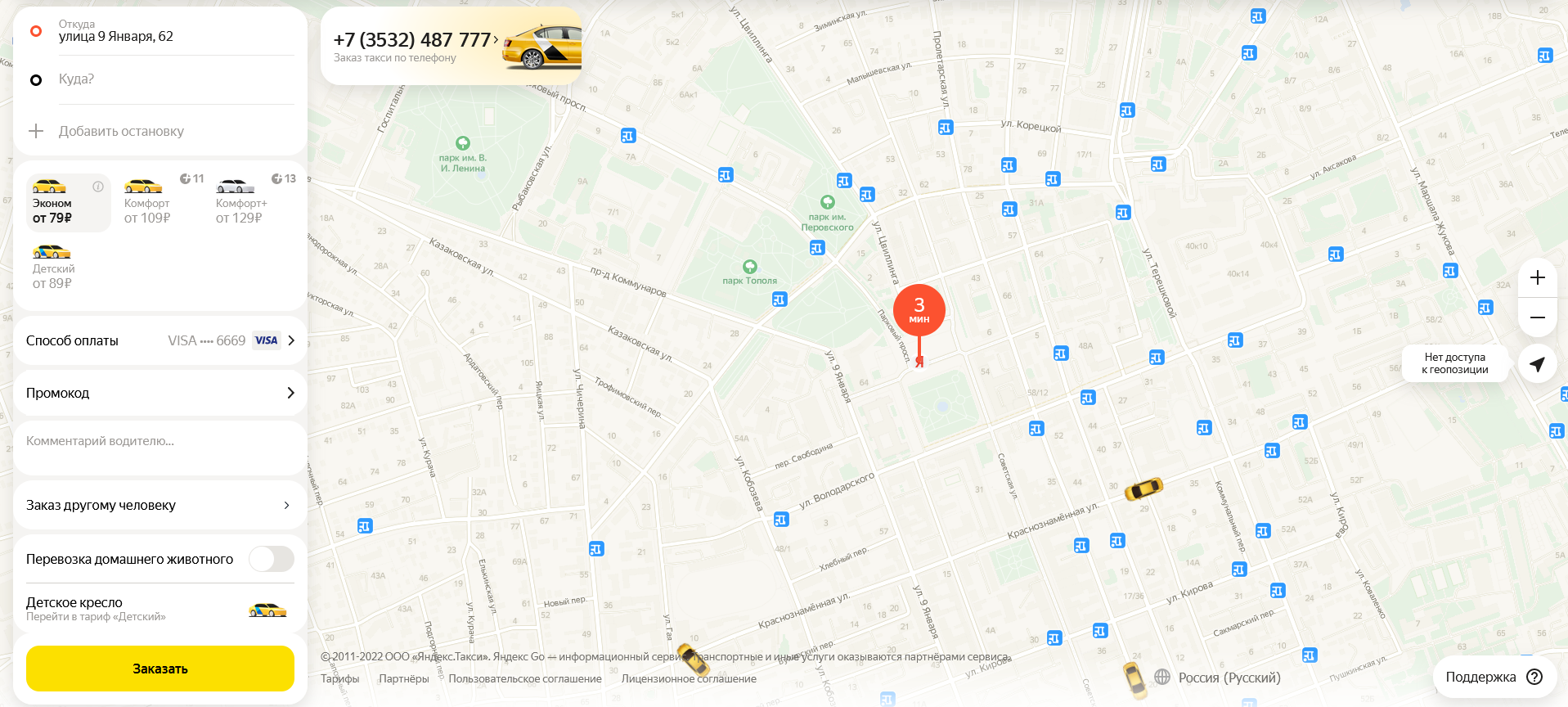


Рисунок 1 – «Яндекс Go»

В данном WEB-приложении можно заказать такси на два или более адреса. Можно выбрать класс машины, удобный для Вас способ оплаты или оставить комментарий водителю.

Достоинства:

1. приятный интерфейс;
2. обширный выбор машин;
3. выбор удобного способа оплаты;
4. возможность заказать по телефону.

Так же в процессе анализа был выделен еще одно WEB-приложение «UBER». На рисунке 2 представлен внешний вид WEB-приложения «UBER».

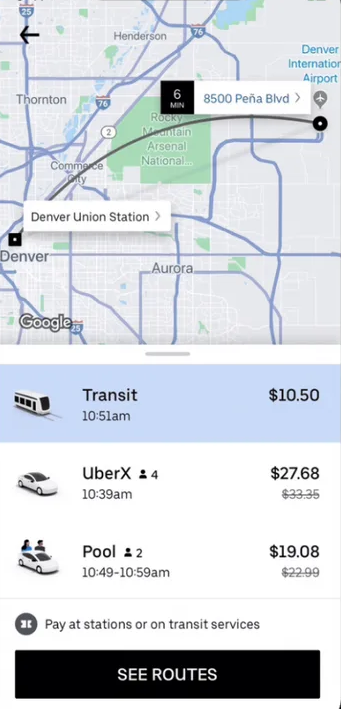


Рисунок 2 – «UBER»

Данное WEB-приложение обладает примерно такими же свойствами что и описанное выше, но уже имеет менее свойств и возможностей, а именно:

* отличный дизайн;
* обширный выбор машин.

На основе анализа существующих интернет магазинов выделим следующие требования к разрабатываемому WEB-приложению:

1. главная страница;
2. наличие автопарка;
3. удобный интерфейс.

# 2 Проектирование базы данных

При проектировании базы данных необходимо определить предметные области. В данной разработке выделено 5 сущностей, описание которых представлено в таблице 1.

Таблица 1 – Сущности

|  |  |
| --- | --- |
| Сущность | Атрибуты |
| Car (Машины) | |  |  | | --- | --- | | Id | Int – Ключ | | color\_Id | Int – Индивидуальный номер цвета | | model\_Id | Int – Индивидуальный номер модели | | statenumder | nvarchar(MAX) – Госномера | |
| Client (Клиенты) | Id int – Ключ  Name nvarchar(MAX) – Имя  L\_Name nvarchar(MAX) – Фамилия  Age int – Возраст  Telephone nvarchar(MAX) – Телефон  Password nvarchar(MAX) – Пароль |
| Color (Цвета) | Id int – Ключ  Name nvarchar(MAX) – Цвет |
| Driver (Водители) | int Id – Ключ  nvarchar(MAX) Name – Имя  nvarchar(MAX) L\_Name – Фамилия  nvarchar(MAX) Telephone – Телефон  nvarchar(MAX) Password – Пароль  nvarchar(MAX) Salary – Зарплата |
| Model (Модели) | int Id – Ключ  nvarchar(MAX) Name – Модель  nvarchar(MAX) Company – Марка |

# 3 Разработка программного приложения

## 3.1 Создание моделей

Одним из ключевых компонентов паттерна MVC являются модели. Ключевая задача моделей – описание структуры и логики используемых данных.

Как правило, все используемые сущности в приложении выделяются в отдельные модели, которые и описывают структуру каждой сущности. В зависимости от задач и предметной области можно выделить различное количество моделей в приложении.

Все модели оформляются как обычные POCO – классы (plain-old CRL objects), то есть обычные классы на языке Создание моделей выполняется по созданным сущностям и по спроектированной схеме базы данных. По данной реализации базы данных создадим пять классов:

1) Класс car(Машины) содержит:

|  |  |
| --- | --- |
| Id | Int – Ключ |
| color\_Id | Int – Индивидуальный номер цвета |
| model\_Id | Int – Индивидуальный номер модели |
| statenumder | nvarchar(MAX) – Госномера |

namespace Taxi.Data.Models

{

public class car

{

public int Id { get; set; }

public int color\_Id { get; set; }

public int model\_Id { get; set; }

public string statenumder { get; set; }

}

}

2) Класс Client (Клиенты) содержит:

Id int – Ключ

Name nvarchar(MAX) – Имя

L\_Name nvarchar(MAX) – Фамилия

Age int – Возраст

Telephone nvarchar(MAX) – Телефон

Password nvarchar(MAX) – Пароль

namespace Taxi.Data.Models

{

public class client

{

public int Id { get; set; }

public string Name { get; set; }

public string L\_Name { get; set; }

public int Age { get; set; }

public string Telephone { get; set; }

public string Password { get; set; }

}

}

3) Класс Color (Цвета) содержит:

Id int – Ключ

Name nvarchar(MAX) – Цвет

namespace Taxi.Data.Models

{

public class color

{

public int Id { get; set; }

public string Name { get; set; }

}

}

4) Класс Driver (Водители) содержит:

int Id – Ключ

nvarchar(MAX) Name – Имя

nvarchar(MAX) L\_Name – Фамилия

nvarchar(MAX) Telephone – Телефон

nvarchar(MAX) Password – Пароль

nvarchar(MAX) Salary – Зарплата

namespace Taxi.Data.Models

{

public class driver

{

public int Id { get; set; }

public string Name { get; set; }

public string L\_Name { get; set; }

public int Age { get; set; }

public string Telephone { get; set; }

public string Password { get; set; }

public string Salary { get; set; }

}

}

5) Класс Model (Модели) содержит:

int Id – Ключ

nvarchar(MAX) Name – Модель

nvarchar(MAX) Company – Марка

namespace Taxi.Data.Models

{

public class model

{

public int Id { get; set; }

public string Name { get; set; }

public string Company { get; set; }

}

}

## 3.2 Создание базы данных

Entity Framework представляет ORM-решение, которое позволяет автоматически связать обычные классы языка C# с таблицами в базе данных. Entity Framework Core нацелен в первую очередь на работу с СУБД MS SQL Server, а так же поддерживает и ряд других СУБД. Одним из подходов в работе с Entity Framework является подход «Code-First». Суть подхода заключается в том, что на основе модели приложения формируется структура базы данных (таблицы, первичные ключи, ссылки), вся эта работа происходит не напрямую с SQL. Обязательным условием класса модели является наличие поля первичного ключа, по умолчанию Entity Framework ищет целочисленное поле, в имени которого присутствует подстрока id, и формирует на его основе первичный ключ.

Класс ApplicationDbContext предназначен для работы с базой данных. Данный класс унаследует все от класса DBContexr, который относится к библиотеке Mictosoft.Entity.FrameworkCore. В классе был создан базовый конструктор по умолчанию и экземпляры функций, которые будут вызываться для работы с данными в базе данных.

namespace Test.Data

{

public class ApplicationDbContext : DbContext

{

public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options) : base(options) {

Database.EnsureCreated();

}

public DbSet<car> car { get; set; }

public DbSet<client> client { get; set; }

public DbSet<color> color { get; set; }

public DbSet<driver> driver { get; set; }

public DbSet<model> model { get; set; }

}

}

Подключения к базе данных выполняется при помощи файла параметров приложения dbsettings с расширением «json». При создании файла создается подключение по умолчанию и необходимо добавить названия базы данных «Taxi».

{

"ConnectionStrings": {

"DefaultConnection": "Server=(localdb)\\MSSQLLocalDB;Database=Taxi;Trusted\_Connection=True;MultipleActiveResultSets=true"

}

}

Для создания самой базы данных надо перейти в консоль диспетчера пакетов, выполнить миграции и обновления базы данных.

## 3.3 Создание контроллеров

Основным элементом в архитектуре ASP.NET Core MVC является контроллер. При получении запроса система маршрутизации выбирает для обработки запроса нужный контроллер и передает ему данные запроса. Контроллер обрабатывает эти данные и посылает обратно результат обработки.

Проект, создаваемый по шаблону ASP.NET Core Web App (Model–View–Controller), по умолчанию содержит как минимум один контроллер – класс HomeController.

В данной работе создан контроллер ColorControlle, который отвечает за отображение на главной странице сайта, информацию о продукции, которая является популярной.

using Microsoft.AspNetCore.Mvc;

using Taxi.Data.Models;

using Taxi.ViewModels;

using Test.Data;

namespace Taxi.Controllers

{

public class ColorController : Controller

{

ApplicationDbContext DB;

public ColorController(ApplicationDbContext context)

{

DB = context;

}

public ActionResult List(){

ColorCarViewModel ColorCar = new ColorCarViewModel();

ColorCar.allcolor = DB.color;

ColorCar.allcar = DB.car;

ColorCar.allmodel = DB.model;

return View(ColorCar);

}

}

}

namespace Test.Controllers

{

public class HomeController : Controller

{

private readonly ILogger<HomeController> \_logger;

public HomeController(ILogger<HomeController> logger)

{

\_logger = logger;

}

public IActionResult Index()

{

return View();

}

public IActionResult Privacy()

{

return View();

}

[ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]

public IActionResult Error()

{

return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });

}

}

}

## 3.4 Создание представлений

В шаблоне MVC (Model–View–Controller, модель – представление – контроллер) представление отвечает за отображение данных приложения и взаимодействие с пользователем. Представление – это HTML-шаблон со встроенной [Razor разметкой](https://docs.microsoft.com/ru-ru/aspnet/core/mvc/views/razor?view=aspnetcore-6.0). Razor разметка – это код, взаимодействующий с разметкой HTML для создания веб–страницы, отправляемой клиенту.

В ASP.NET Core MVC представления – это .cshtml файлы, использующие [язык программирования C#](https://docs.microsoft.com/ru-ru/dotnet/csharp/) в Razor разметке. Как правило, файлы представлений объединяются в папки с именами, соответствующими отдельным [контроллерам](https://docs.microsoft.com/ru-ru/aspnet/core/mvc/controllers/actions?view=aspnetcore-6.0) приложения. Папки хранятся в Views-папке в корне приложения. Создадим мастер-страницу, которая будет называться «\_Layout». Она нужна для того, чтобы не повторять код.

В программе создано следующее представление:

ColorCarViewModel

using System.Collections.Generic;

using Taxi.Data.Models;

namespace Taxi.ViewModels

{

public class ColorCarViewModel

{

public IEnumerable<color> allcolor { get; set; }

public IEnumerable<car> allcar { get; set; }

public IEnumerable<model> allmodel { get; set; }

}

}

# 4 Тестирование созданного программного приложения

При открытии проекта открывается главная страница такси. На рисунке 3 изображена главная страница.

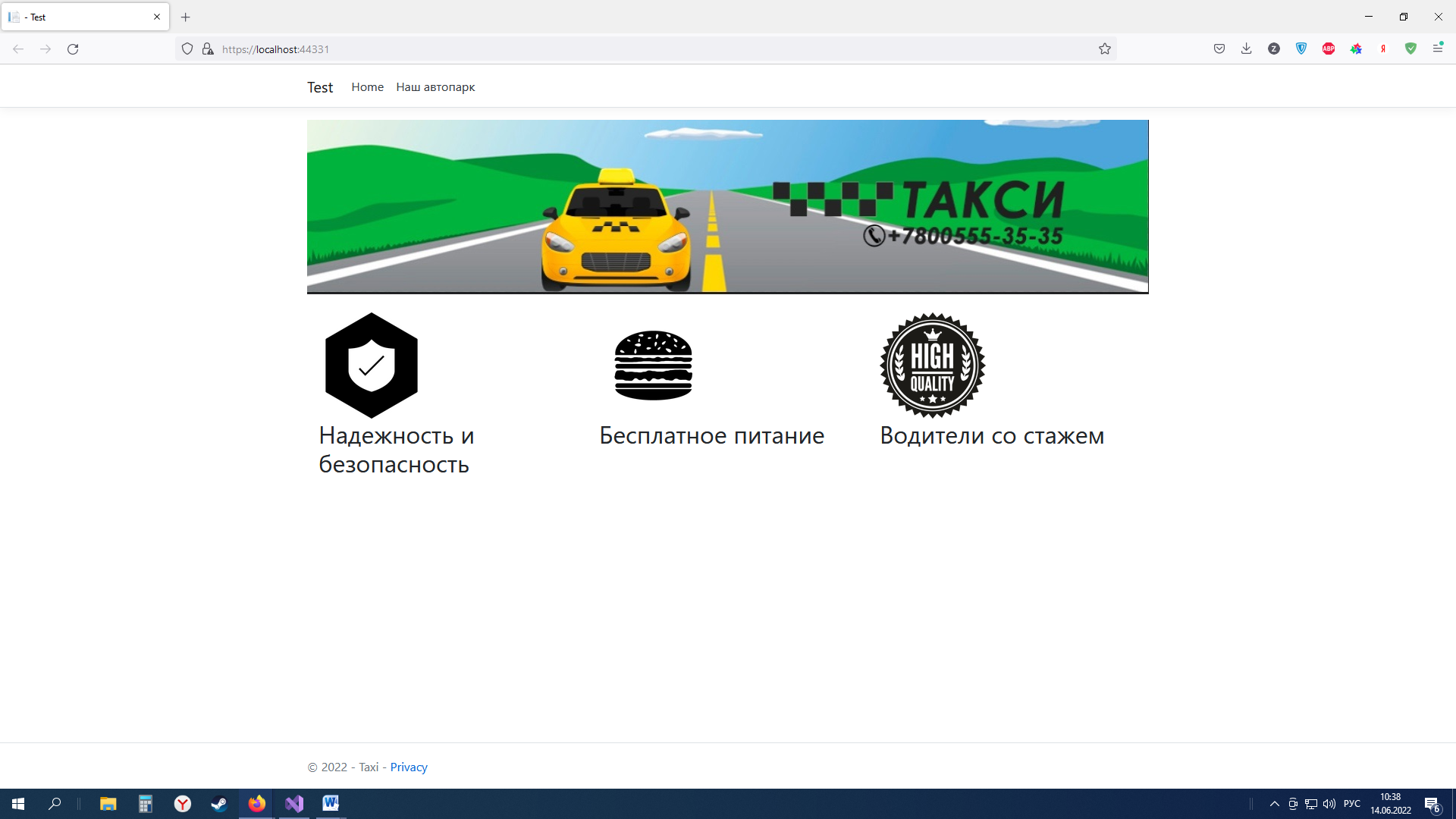


Рисунок 3 – Главная страница

При нажатии кнопки «Наш автопарк» Вы можете увидеть список машин, их цвета, модель и марку. На рисунке 4 изображена страница «Наш автопарк».

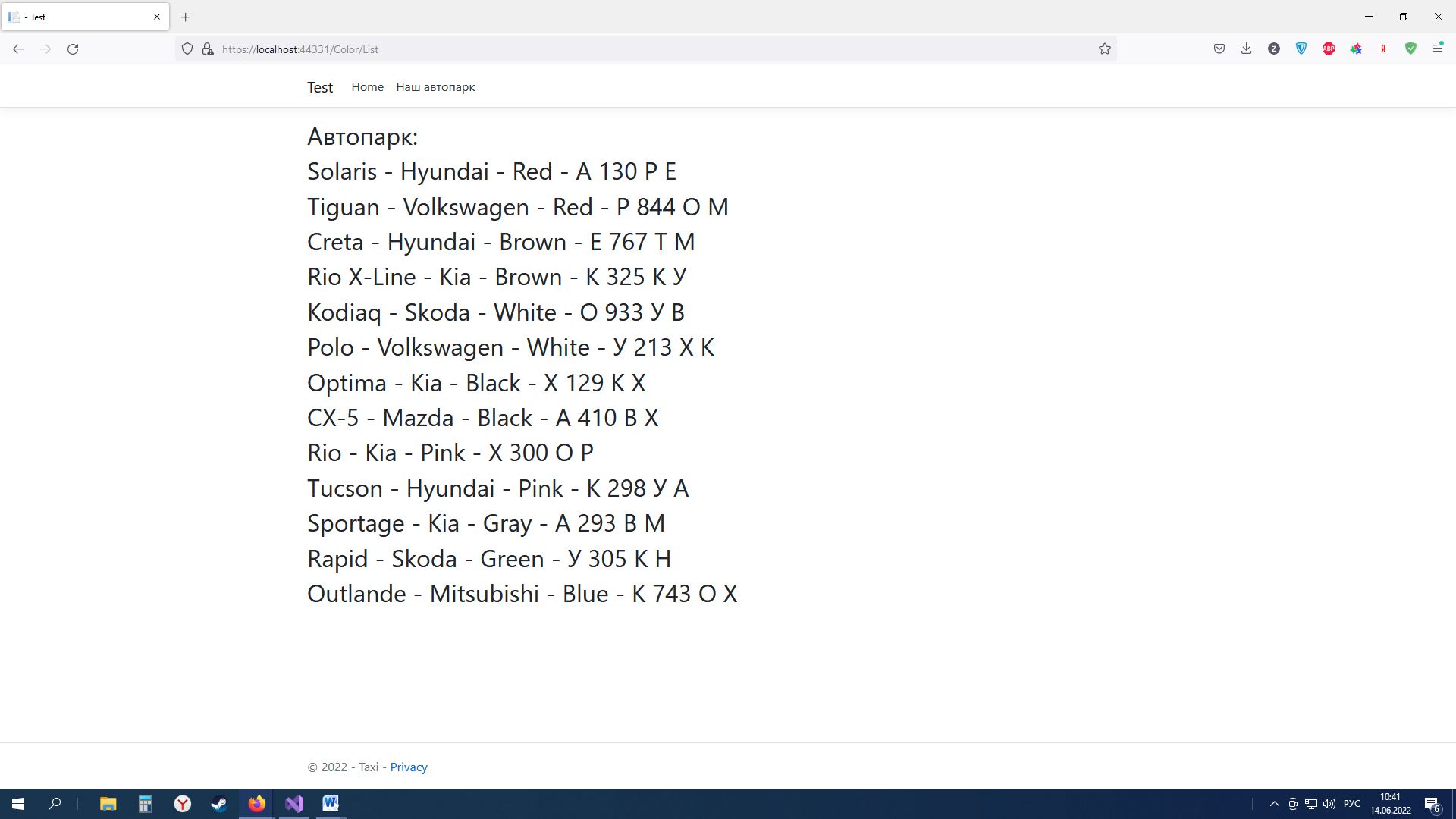


Рисунок 4 – Страница «Наш автопарк»

# Заключение

В результате выполнения курсовой работы создано программное приложение для заказа такси в интернете. Для этого изучена литература по разработке сайтов и работе с базами данных. Проанализированы существующие программные реализации и сформулированы требования к разрабатываемой информационной системе. Выбрана среда разработки Visual Studio с технологией ASP.NET Core как наиболее удобный для создания сайтов. Разработан интерфейс и функционал. Продукт был протестирован на наличие ошибок, а также на одногруппниках и преподавателе.

# Список использованных источников

1. Яндекс.Такси [Электронный ресурс]. – Режим доступа: <https://taxi.yandex.ru/> – 30.05.2022.
2. Введение в ASP.NET Core [Электронный ресурс]. – Режим доступа: <https://metanit.com/sharp/aspnet6/1.1.php> – 30.05.2022.
3. Введение в ASP.NET Core MVC 5 [Электронный ресурс]. – Режим доступа: https://metanit.com/sharp/mvc5/1.1.php – 30.05.2022.
4. Добавление Identity в проект [Электронный ресурс]. – Режим доступа: <https://metanit.com/sharp/aspnet5/16.2.php> – 30.05.2022.
5. Введение в Entity Framework Core [Электронный ресурс]. – Режим доступа: <https://metanit.com/sharp/efcore/1.1.php> – 30.05.2022.