Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего образования «Ижевский государственный технический университет
имени М.Т. Калашникова»
(ФГБОУ ВО «ИжГТУ имени М.Т. Калашникова»)

Институт «Информатика и вычислительная техника»
Кафедра «Вычислительная техника»

Отчёт
по лабораторной работе №3
по дисциплине
«Объектно-ориентированное программирование»

по теме

«Проектирование приложений, SOLID, паттерны проектирования»

Выполнил: студент гр. Б04-780-2
Контробаев Р.А.

27.05.2020

Проверил: к.т.н., доцент каф. ВТ
Марков Е.М.

Ижевск 2020

Лабораторная работа №3

«Проектирование приложений, SOLID, паттерны проектирования»

Цель работы

Получить практические навыки проектирования приложений, на примере веб-приложения.

Постановка задачи

1. Разработать веб-приложение по заданию на основе ASP.NET или любой другой, с использованием MVC;
2. Приложение должно содержать страницу с перечнем созданных записей (объектов) и, хотя бы одно действие для каждого из них, например, «подробнее», «купить» или подобное, открывающее другую страницу.
3. (Дополнительно, не обязательно для выполнения) Хранить данные о записях (объектах) в базе данных.

Описание использованных технологий

Model-View-Controller (MVC, «Модель-Представление-Контроллер», «Модель-Вид-Контроллер») — схема разделения данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента: модель, представление и контроллер — таким образом, что модификация каждого компонента может осуществляться независимо.

- Модель (Model) предоставляет данные и реагирует на команды контроллера, изменяя своё состояние.
- Представление (View) отвечает за отображение данных модели пользователю, реагируя на изменения модели.
- Контроллер (Controller) интерпретирует действия пользователя, оповещая модель о необходимости изменений.

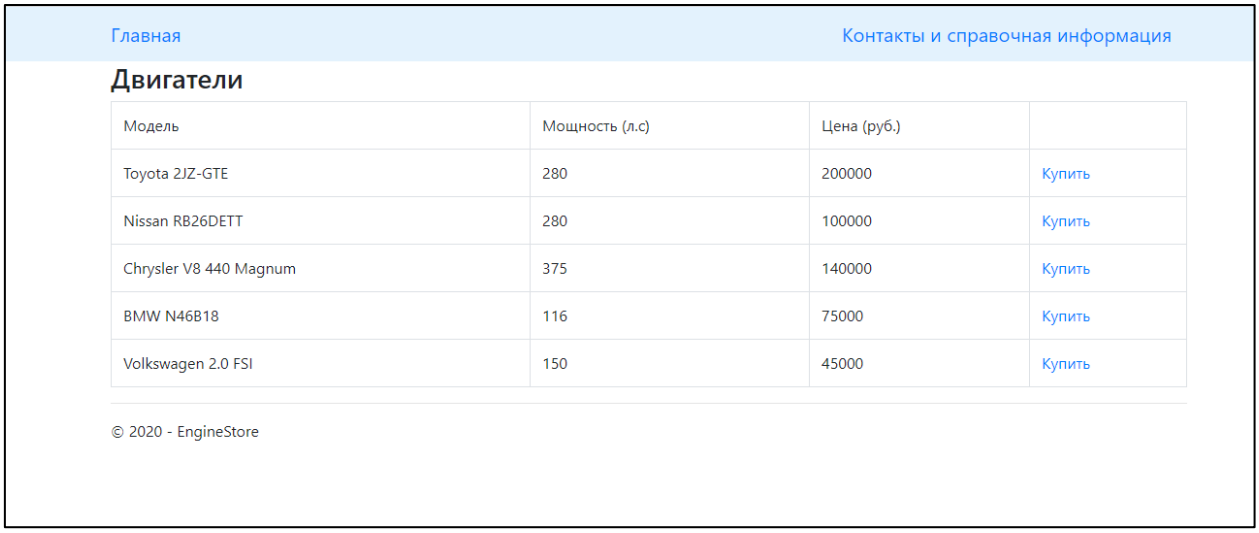
ASP.NET CORE – свободно-распространяемый кросс-платформенный фреймворк для создания веб-приложений с открытым исходным кодом. Данная платформа разрабатывается компанией Майкрософт совместно с сообществом и имеет большую производительность по сравнению с ASP.NET. Имеет модульную структуру и совместима с такими операционными системами как Windows, Linux и macOS.

HTML (от англ. HyperText Markup Language — «язык гипертекстовой разметки») — стандартизированный язык разметки документов в сети Интернет. Большинство веб-страниц содержат описание разметки на языке HTML (или XHTML). Язык HTML интерпретируется браузерами; полученный в результате интерпретации форматированный текст отображается на экране монитора компьютера или мобильного устройства.

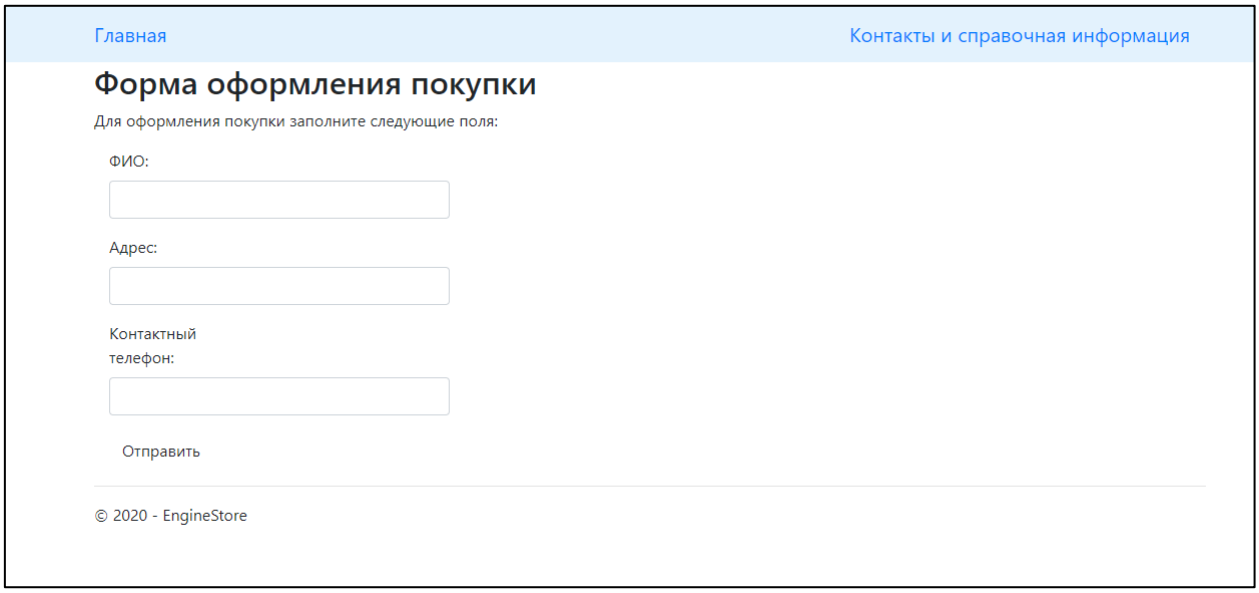
CSS (Cascading Style Sheets — каскадные таблицы стилей) — формальный язык описания внешнего вида документа, написанного с использованием языка разметки. Преимущественно используется как средство описания, оформления внешнего вида веб-страниц, написанных с помощью языков разметки HTML и XHTML, но может также применяться к любым XML-документам, например, к SVG или XUL.

MySQL – это одна из самых популярных и самых распространенных СУБД (система управления базами данных) в сети Интернет. Она не предназначена для работы с большими объемами информации, но ее применение идеально для интернет сайтов, как небольших, так и достаточно крупных.

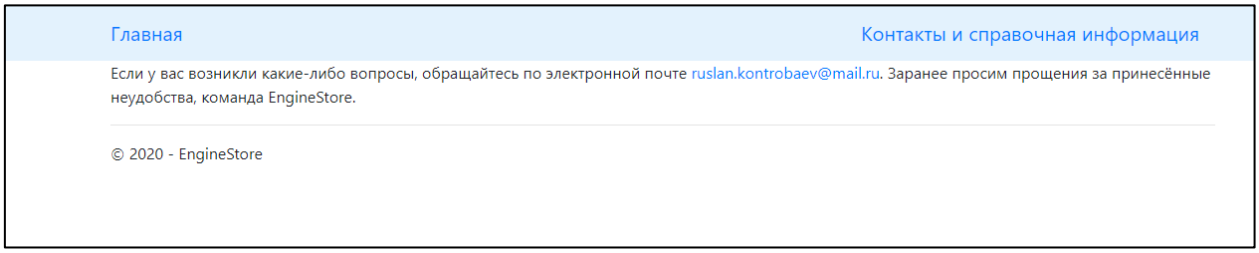
Скриншоты приложения



Скриншот 1



Скриншот 2



Скриншот 3

Листинг демонстрационной программы

appsettings.json

```
{
  "ConnectionStrings":
  {
    "DefaultConnection":
"Server=(localdb)\\mssqllocaldb;Database=enginestoredb;Trusted_Connection=Tru
e;MultipleActiveResultSets=true"
  },
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "Information"
    }
  },
  "AllowedHosts": "*"
}
```

Program.cs

```
using System;
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using WebApplication1.Models;

namespace WebApplication1
{
    public class Program
    {
        public static void Main(string[] args)
        {
            var host = CreateHostBuilder(args).Build();
            using (var scope = host.Services.CreateScope())
```

```
            {
                var services = scope.ServiceProvider;
                try
                {
                    var                         context                         =
services.GetRequiredService<EngineContext>();
                    SampleData.Initialize(context);
                }
                catch (Exception ex)
                {
                    var                         logger                          =
services.GetRequiredService<ILogger<Program>>();
                    logger.LogError(ex, "An error occurred seeding the DB.");
                }
            }
            host.Run();
        }
        public static IHostBuilder CreateHostBuilder(string[] args) =>
            Host.CreateDefaultBuilder(args)
                .ConfigureWebHostDefaults(webBuilder =>
                {
                    webBuilder.UseStartup<Startup>();
                });
    }
}
```

SampleData.cs

```
using System.Linq;
using WebApplication1.Models;

namespace WebApplication1
{
    public class SampleData
    {
        public static void Initialize(EngineContext context)
        {
            if (!context.Engines.Any())
```

```
    {
        context.Engines.AddRange
        (
            new Engine
            {
                Name = "Toyota 2JZ-GTE",
                Power = 280,
                Price = 200000.0
            },
            new Engine
            {
                Name = "Nissan RB26DETT",
                Power = 280,
                Price = 100000.0
            },
            new Engine
            {
                Name = "Chrysler V8 440 Magnum",
                Power = 375,
                Price = 140000.0
            },
            new Engine
            {
                Name = "BMW N46B18",
                Power = 116,
                Price = 75000.0
            },
            new Engine
            {
                Name = "Volkswagen 2.0 FSI",
                Power = 150,
                Price = 45000.0
            }
        );
        context.SaveChanges();
    }
}
```

```
    }
}
```

Startup.cs

```
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using WebApplication1.Models;

namespace WebApplication1
{
    public class Startup
    {
        public Startup(IConfiguration configuration)
        {
            Configuration = configuration;
        }

        public IConfiguration Configuration { get; }

        public void ConfigureServices(IServiceCollection services)
        {
            services.AddMvc();
            string                          connection                          =
Configuration.GetConnectionString("DefaultConnection");
            services.AddDbContext<EngineContext>(options                        =>
options.UseSqlServer(connection));
            services.AddControllersWithViews();
        }

        public  void  Configure(IApplicationBuilder  app,  IWebHostEnvironment
env)
        {
            if (env.IsDevelopment())
```

```
            {
                app.UseDeveloperExceptionPage();
            }
            else
            {
                app.UseExceptionHandler("/Home/Error");
                app.UseHsts();
            }
            app.UseHttpsRedirection();
            app.UseStaticFiles();

            app.UseRouting();

            app.UseAuthorization();

            app.UseEndpoints(endpoints =>
            {
                endpoints.MapControllerRoute(
                    name: "default",
                    pattern: "{controller=Home}/{action=Index}/{id?}");
            });
        }
    }
}
```

HomeController.cs

```
using System.Linq;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;

namespace WebApplication1.Controllers
{
    public class HomeController : Controller
    {
        EngineContext db;
        public HomeController(EngineContext context)
        {
```

```
            db = context;
        }

        public IActionResult Index()
        {
            return View(db.Engines.ToList());
        }

        [HttpGet]
        public IActionResult Buy(int? id)
        {
            if (id == null) return RedirectToAction("Index");
            ViewBag.EngineId = id;
            return View();
        }

        [HttpPost]
        public string Buy(Order order)
        {
            db.Orders.Add(order);
            db.SaveChanges();// Сохраняем в бд все изменения
            return order.User + " спасибо за покупку в нашем магазине. Ваш
заказ оформлен, рады будем видеть вас вновь!";
        }

        public IActionResult Info()
        {
            return View();
        }
    }
}
```

Engine.cs

```
namespace WebApplication1.Models
{
    public class Engine
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public int Power { get; set; }
        public double Price { get; set; }
    }
}
```

EngineContext.cs

```
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using static Microsoft.EntityFrameworkCore.DbLoggerCategory;

namespace WebApplication1.Models
{
    public class EngineContext : DbContext
    {
        public DbSet<Engine> Engines { get; set; }
        public DbSet<Order> Orders { get; set; }

        public EngineContext(DbContextOptions<EngineContext> options)
            : base(options)
        {
            Database.EnsureCreated(); // Создание БД, если уже есть, то не
создаёт
        }
    }
}
```

Order.cs

```
namespace WebApplication1.Models
{
    public class Order
    {
        public int OrderId { get; set; }
        public string User { get; set; } // имя фамилия покупателя
        public string Address { get; set; } // адрес покупателя
        public string ContactPhone { get; set; } // контактный телефон покупателя

        public int EngineId { get; set; } // ссылка на связанную модель Engine
        public Engine Engine { get; set; }
    }
}
```

Buy.cshtml

```
@{
    ViewData["Title"] = "Оформление заказа";
}
<h2>Форма оформления покупки</h2>

<form method="post" class="form-horizontal" role="form">
    <input type="hidden" value="@ViewBag.EngineId" name="EngineId" />
    <p>Для оформления покупки заполните следующие поля:</p>

    <div class="form-group">
        <label for="User" class="col-md-2 control-label">ФИО:</label>
        <div class="col-md-4">
            <input type="text" name="User" class="form-control" />
        </div>
    </div>

    <div class="form-group">
        <label for="Address" class="col-md-2 control-label">Адрес:</label>
```

```
        <div class="col-md-4">
            <input type="text" name="Address" class="form-control" />
        </div>
    </div>

    <div class="form-group">
        <label class="col-md-2 control-label">Контактный телефон:</label>
        <div class="col-md-4">
            <input type="text" name="ContactPhone" class="form-control" />
        </div>
    </div>

    <div class="form-group">
        <div class="col-md-offset-2 col-md-10">
            <input type="submit" class="btn btn-default" value="Отправить" />
        </div>
    </div>

</form>
```

Index.cshtml

```
@model IEnumerable<WebApplication1.Models.Engine>
@{
    ViewData["Title"] = "Каталог товаров";
}
<h3>Двигатели</h3>
<table class="table table-bordered">
    <tr>
        <td>Модель</td>
        <td>Мощность (л.с)</td>
        <td>Цена (руб.)</td>
        <td></td>
    </tr>
    @foreach (var engine in Model)
    {
        <tr>
```

```
            <td>@engine.Name</td>
            <td>@engine.Power</td>
            <td>@engine.Price</td>
            <td><a href="~/Home/Buy/@engine.Id">Купить</a></td>
        </tr>
    }
</table>
```

Info.cshtml

```
@{
    ViewData["Title"] = "Справочная информация";
}
<h>Если у вас возникли какие-либо вопросы, обращайтесь по электронной почте <a  href="https://e.mail.ru/compose/">ruslan.kontrobaev@mail.ru</a>.  Заранее просим прощения за принесённые неудобства, команда EngineStore.</h>
```

_Layout.cshtml

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>@ViewData["Title"]</title>
    <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.css" />
</head>
<body>
    <!--Шапка-->
    <nav class="navbar navbar-inverse" style="background-color: #e3f2fd;">
        <div class="container">
            <div>
                <ul class="nav navbar-nav">
                    <li><a          href="~/Home/Index"          class="navbar-brand">Главная</a></li>
                </ul>
            </div>
```

```
            <div>
                <ul class="nav navbar-nav">
                    <li><a href="~/Home/Info" class="navbar-brand">Контакты и
справочная информация</a></li>
                </ul>
            </div>
        </div>
    </nav>
    <!--основной контент-->
    <div class="container body-content">
        @RenderBody()
        <hr />
        <footer>
            <p>© 2020 - EngineStore</p>
        </footer>
    </div>
</body>
</html>
```

Вывод

В результате выполнения лабораторной работы были освоены практические навыки проектирования приложений, на примере веб-приложения. Было разработано веб-приложение по заданию на основе ASP.NET Core с использованием MVC с хранением записей (объектов) в базе данных.