Министерство науки и высшего образования Российской Федерации

ФГАОУ ВО «Уральский федеральный университет

имени первого Президента России Б.Н. Ельцина»

Институт новых материалов и технологий

Кафедра «Теплофизика и информатика в металлургии»

**Сборка и запуск контейнера в Docker**

**ОТЧЕТ**

**по практической работе № 1**

**по дисциплине «Основы методологии Development Operation»**

Направление 09.03.02 «Информационные системы и технологии» (уровень бакалавриата)

Образовательная программа

09.03.02/33.02 «Информационные системы и технологии» (СУОС)

Вариант №09

Студент

группы НМТ-413901 Я.В.Крашенинников

Преподаватель:

профессор, д.т.н. В.В.Лавров

Екатеринбург

2024

Оглавление

[1.1 Цель работы 3](#_Toc181371125)

[1.2 Ход проведения работы 3](#_Toc181371126)

[1.2.1 Разработка тестового приложения Visual Studio .NET Core 3](#_Toc181371127)

[1.2.2 Создание контейнера с использованием Docker 4](#_Toc181371128)

[1.2.3 Размещение приложения в системе контроля версий GitHub 5](#_Toc181371129)

[1.2.4 Сборка образа с использованием команды docker build 5](#_Toc181371130)

[1.2.5 Создание контейнера с использованием Docker-образа и запуск 5](#_Toc181371131)

[1.2.6 Демонстрация возможности публикации и остановки контейнера 6](#_Toc181371132)

[1.3 Выводы 7](#_Toc181371133)

[Приложение А Листинг программного кода приложения «Калькулятор» 8](#_Toc181371134)

[(контроллер) 8](#_Toc181371135)

[Приложение Б Листинг программного кода приложения «Калькулятор» (appsettings.json) 9](#_Toc181371136)

[Приложение В Листинг программного кода приложения «Калькулятор» (appsettings.Development.json) 10](#_Toc181371137)

[Приложение Г Листинг программного кода приложения «Калькулятор» (Program.cs) 11](#_Toc181371138)

**Практическая работа «Сборка и запуск контейнера в Docker»**

## 1.1 Цель работы

Цель работы состоит в настройке и выполнении сборки контейнера в Docker с использованием PuTTY для удаленного доступа к серверу или машине в локальной сети. Основная задача заключается в демонстрации процесса сборки Docker-контейнера на удаленной машине через SSH-подключение с помощью PuTTY. После завершения действий целью является успешная сборка и запуск контейнера в Docker на удаленной машине в локальной сети, обеспечивая тем самым возможность управления и тестирования приложения в изолированной среде на удаленной платформе.

## 1.2 Ход проведения работы

### 1.2.1 Разработка тестового приложения Visual Studio .NET Core

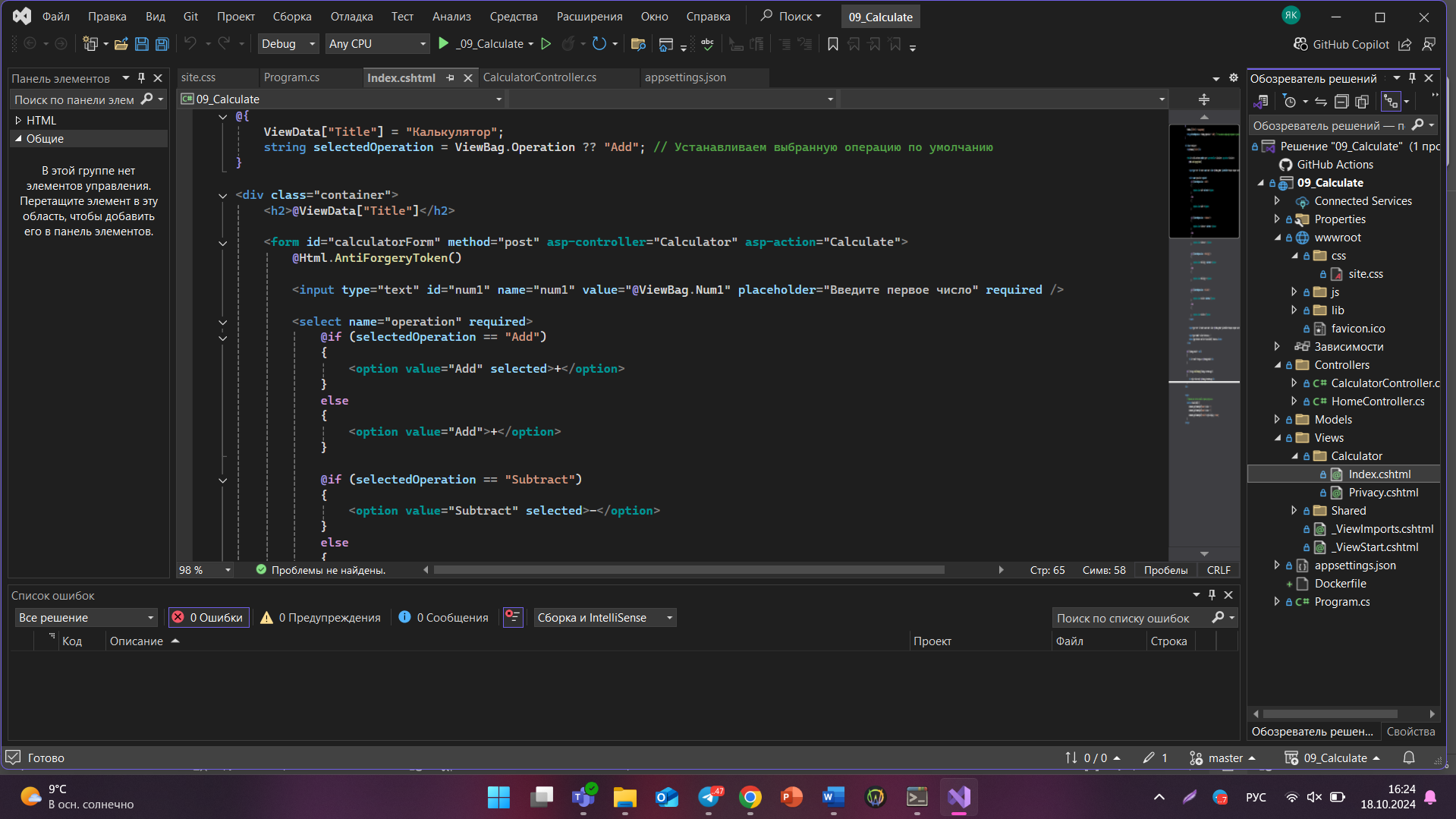


Рисунок 1 – Создание MVC приложения в MVS2022

### 1.2.2 Создание контейнера с использованием Docker

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 2 – Создание докерфайла в корневой папке проекта

Листинг кода:

FROM mcr.microsoft.com/dotnet/aspnet:6.0 AS base

WORKDIR /app

FROM mcr.microsoft.com/dotnet/sdk:6.0 AS build

COPY . /src

WORKDIR /src

RUN ls

RUN dotnet restore

RUN dotnet build "./09\_Calculate.csproj" -c Release -o /app/build

FROM build AS publish

RUN dotnet publish "./09\_Calculate.csproj" -c Release -o /app/publish

FROM base AS final

WORKDIR /app

COPY --from=publish /app/publish ./

ENTRYPOINT ["dotnet", "09\_Calculate.dll"]

### 1.2.3 Размещение приложения в системе контроля версий GitHub

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 3 -Опубликованный публичный проект на моем Гите

Ссылка - https://github.com/krosshiik/09\_Calculate.git

### 1.2.4 Сборка образа с использованием команды docker build

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Рисунок 4 – Создание контейнера с помощью команды sudo docker build

### 1.2.5 Создание контейнера с использованием Docker-образа и запуск

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 5 – Docker images, мой созданный контейнер

### 1.2.6 Демонстрация возможности публикации и остановки контейнера

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 6 – Введенные команды старт стоп

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок 7 – Работающий проект на нужном IP

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 8 – Неработающее приложение после остановки контейнера через Putty

## 1.3 Выводы

В ходе работы с PuTTY в локальной сети УРФУ и выполнения действий по сборке и запуску контейнеров в Docker мне удалось изучить процесс удаленного доступа через SSH с помощью PuTTY, а также освоить основы контейниризации приложений с использованием Docker. Это полезные навыки для разработчиков и администраторов, позволяющие управлять удаленными серверами, контейнеризировать приложения для удобного развертывания и масштабирования. В ходе работы я настраивал и устанавливал PuTTY для удаленного доступа и создавал, собирал и запускал контейнеры в Docker для изоляции приложений и обеспечения их портабельности и удобства управления. Эти навыки помогут мне эффективно управлять и развивать приложения на серверах в локальной сети у заказчика.

## Приложение А Листинг программного кода приложения «Калькулятор»

## (контроллер)

using Microsoft.AspNetCore.Mvc;

namespace \_09\_Calculate.Controllers

{

public enum Operation { Add, Subtract, Multiply, Divide }

public class CalculatorController : Controller

{

[HttpGet]

public IActionResult Index()

{

return View();

}

[HttpPost]

[ValidateAntiForgeryToken]

public IActionResult Calculate(double num1, double num2, Operation operation)

{

double result = 0;

string errorMessage = null;

try

{

switch (operation)

{

case Operation.Add:

result = num1 + num2;

break;

case Operation.Subtract:

result = num1 - num2;

break;

case Operation.Multiply:

result = num1 \* num2;

break;

case Operation.Divide:

if (num2 != 0)

{

result = num1 / num2;

}

else

{

errorMessage = "Ошибка: деление на ноль невозможно.";

}

break;

}

}

catch (Exception ex)

{

errorMessage = "Произошла ошибка: " + ex.Message;

}

ViewBag.Result = result;

ViewBag.Num1 = num1; // Сохраняем первое число

ViewBag.Num2 = num2; // Сохраняем второе число

ViewBag.Operation = operation.ToString(); // Сохраняем выбранную операцию

ViewBag.ErrorMessage = errorMessage; // Передаем сообщение об ошибке, если оно есть

return View("Index");

}

}

}

## Приложение Б Листинг программного кода приложения «Калькулятор» (appsettings.json)

{

"Kestrel": {

"Endpoints": {

"Http": {

"Url": "http://0.0.0.0:5009" // 09 – номер моего варианта

}

}

},

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

},

"AllowedHosts": "\*"

}

## Приложение В Листинг программного кода приложения «Калькулятор» (appsettings.Development.json)

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft": "Warning",

"Microsoft.Hosting.Lifetime": "Information"

## Приложение Г Листинг программного кода приложения «Калькулятор» (Program.cs)

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllersWithViews();

var app = builder.Build();

// Configure the HTTP request pipeline.

if (!app.Environment.IsDevelopment())

{

app.UseExceptionHandler("/Home/Error");

// The default HSTS value is 30 days. You may want to change this for production scenarios, see https://aka.ms/aspnetcore-hsts.

app.UseHsts();

}

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseRouting();

app.UseAuthorization();

app.MapControllerRoute(

name: "default",

pattern: "{controller=Calculator}/{action=Index}/{id?}");

app.Run();