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»

**Управление контейнерами с помощью Docker Compose**

**и использование готовых образов**

**ОТЧЕТ**

**по практической работе № 2**

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

**Вариант №9**

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

Образовательная программа

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

Студент

группы НМТ-413901 Я.В.Крашенинников

Преподаватель:

профессор, д.т.н. В.В.Лавров

Екатеринбург

2024

Оглавление

[1.1 Цель работы 3](#_Toc181399945)

[1.2 Ход проведения работы 3](#_Toc181399946)

[1.2.1 Разработка тестового приложения Visual Studio .NET Core 4](#_Toc181399947)

[1.2.2 Создание конфигурационного файла docker-compose.yml 5](#_Toc181399948)

[1.2.3 Размещение приложения в системе контроля версий GitHub 6](#_Toc181399949)

[1.2.4 Сборка образов с использованием команды docker compose 6](#_Toc181399950)

[1.2.5 Установка программы HeidiSQL для сопровождения базы данных в СУБД MariaDB 7](#_Toc181399951)

[1.2.6 Настройка миграции базы данных в MariaDB 8](#_Toc181399952)

[1.2.7 Демонстрация работы приложения с базой данных в СУБД MariaDB 9](#_Toc181399953)

[1.3 Выводы 10](#_Toc181399954)

[Приложение А Листинг программного кода приложения «Калькулятор» (контроллер) 11](#_Toc181399955)

[Приложение Б Листинг программного кода приложения «Калькулятор» (appsettings.json) 13](#_Toc181399956)

[Приложение В Листинг программного кода приложения «Калькулятор» (appsettings.Development.json) 14](#_Toc181399957)

[Приложение Г Листинг программного кода приложения «Калькулятор» (Program.cs) 15](#_Toc181399958)



**Практическая работа «Управление контейнерами**

**с помощью Docker Compose и использование готовых образов»**

## 1.1 Цель работы

Разработать тестовое приложение «Калькулятор» на платформе .NET Core, включающее функционал для хранения всех выполненных операций в базе данных MariaDB и настроить взаимодействие с этой базой данных через Entity Framework. В процессе требуется создать и настроить многоконтейнерное приложение с использованием Docker для упрощения развёртывания и обеспечения независимости окружения. Итоговый проект должен быть размещён в системе контроля версий GitHub, база данных — доступна для администрирования с помощью программы HeidiSQL.

## 1.2 Ход проведения работы

Для начала установил все необходимые пакеты с указанными версиями в методичке:

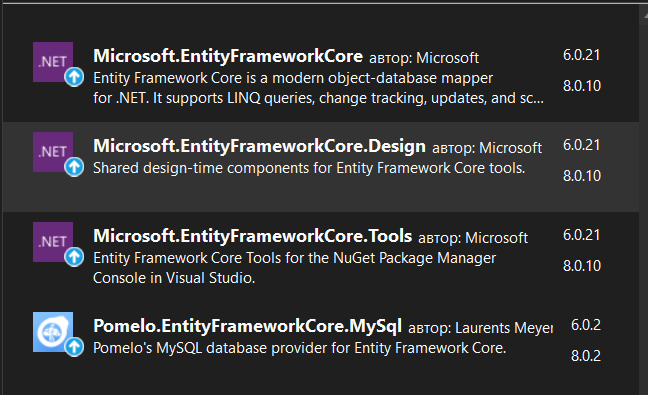


Рисунок 1 – Установка пакетов NuGet

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

Для начала я добавил все недостающие части кода, создал папку Data, где описал все переменные и настроил подключение к БД:

Изображение выглядит как текст, программное обеспечение, Мультимедийное программное обеспечение, Графическое программное обеспечение

Автоматически созданное описание

Рисунок 2 – созданная папка Data

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

Автоматически созданное описание

Рисунок 3 – Описание ConnectionStrings в appsetting.json

На этом моменте главным было правильно указать названия в DefaultConnection.

Также необходимо настроить appsetting.Devol-nt.json, чтобы успешно подключиться на нужном порте и IP к БД:

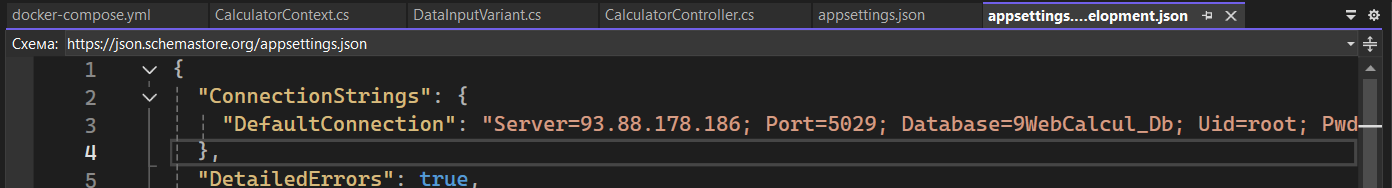


Рисунок 4 - Описание ConnectionStrings в appsetting.Devolopment

### 1.2.2 Создание конфигурационного файла docker-compose.yml

version: '3.7'

services:

mariadb:

container\_name: web-mariadb\_09

hostname: mariadb

image: mariadb:10.5

restart: always

environment:

MARIADB\_ROOT\_PASSWORD: password

MARIADB\_DATABASE: 9WebCalcul\_Db # Имя базы данных, указанное в настройках подключения

MARIADB\_USER: root # Имя пользователя, указанное в настройках подключения

MARIADB\_PASSWORD: password # Пароль, указанный в настройках подключения

ports:

- "5029:3306" # Используем стандартный порт MariaDB 3306

web:

container\_name: web-app-09calculate

hostname: app

build: ./

ports:

- "5009:5009" # Порт, использующийся для веб-приложения

depends\_on:

- mariadb

Тут:

mariadb: Создаёт контейнер для базы данных MariaDB версии 10.5 с именем web-mariadb\_09. Указываются параметры подключения: имя базы данных, пользователя и пароль, а также прокидывается порт 5029 на хосте к стандартному порту 3306 в контейнере.

web: Создает контейнер для веб-приложений с именем web-app-09calculate, компилируя приложение из текущего каталога ( ./). Портируется 5009 в контейнер и перекидывается на 5009 хост. Контейнер зависит от данных базы данных mariadb, поэтому база данных запускается при запуске веб-приложений.

### 1.2.3 Размещение приложения в системе контроля версий GitHub

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

Автоматически созданное описание

Рисунок 5 – Размещенный файл на GitHub в директории проекта

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

### 1.2.4 Сборка образов с использованием команды docker compose

В процессе выполнения я столкнулся с проблемой – при выполнении docker compose у меня дублировались контейнеры на :5009 порте, в следствии чего калькулятор запускался, но не было связи с БД. Для удаления контейнера из первой практической нужно было ввести «*docker ps -a*», скопировать ID старого контейнера, остановить его «*docker stop “ID”*» и удалить «*docker rm “ID”*».

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

Автоматически созданное описание

Рисунок 6 – docker compose up -d, запускаются все контейнеры

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

Автоматически созданное описание

Рисунок 7 – Если всё в статусе UP, значит будет работать

После запуска/перезапуска compose требуется обновить БД в visual studio.

### 1.2.5 Установка программы HeidiSQL для сопровождения базы данных в СУБД MariaDB

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

Автоматически созданное описание

Рисунок 8 – Подключение к БД по указанным параметрам

HeidiSQL была установлена стандартно, при установке пользователь – root, пароль – password. Порт указан 5029 (номер варианта+20), IP сервера, эти данные взяты из кофигурации *appsetting*

### 1.2.6 Настройка миграции базы данных в MariaDB

Вызвал консоль диспетчера пакетов и выполнил команду: *add-migration m\_init*

В проекте создана папка Migrations с двумя файлами.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 9 – Миграция БД

Далее выполняю обновление базы данных с помощью команды: *update-database*. Обновлять БД нужно каждый раз перед новой сессией (как я понял)

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

Автоматически созданное описание

Рисунок 10 – Done означает что у нас все поучилось и данные будут улетать в БД

### 1.2.7 Демонстрация работы приложения с базой данных в СУБД MariaDB

Изображение выглядит как текст, программное обеспечение, Значок на компьютере, Шрифт

Автоматически созданное описание

Рисунок 11 – Моя БД с сохраненным данными на сервере и локалхосте

Изображение выглядит как текст, снимок экрана, программное обеспечение, веб-страница

Автоматически созданное описание

Рисунок 12 – Расчет на сервере

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

Автоматически созданное описание

Рисунок 13 – Расчёт на localhost

## 1.3 Выводы

В ходе практической работы был разработан тестовый проект «Калькулятор», расширенный функционал для учета всех операций в объектах данных MariaDB. С помощью Entity Framework были созданы приложения и успешно реализована настройка взаимодействия приложения с базой данных, что обеспечивает автоматическое сохранение вычислений. Настройка многоконтейнерного окружения через Docker упростила развертывание и настройку компонентов изоляции, использование GitHub упростило контрольную версию и резервное хранение проекта. Итоговая проверка подтверждает корректность работы приложения и базы данных, а также успешную интеграцию всех необходимых сервисов.

## Приложение А Листинг программного кода приложения «Калькулятор» (контроллер)

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using \_09\_Calculate.Data;

namespace \_09\_Calculate.Controllers

{

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

public class CalculatorController : Controller

{

private CalculatorContext \_context;

public CalculatorController(CalculatorContext context)

{

\_context = context;

}

[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;

ViewBag.Num2 = num2;

ViewBag.Operation = operation.ToString();

ViewBag.ErrorMessage = errorMessage;

// Создаем экземпляр класса DataInputVariant и заполняем его

var dataInputVariant = new DataInputVariant

{

Operand\_1 = num1.ToString(),

Operand\_2 = num2.ToString(),

Type\_operation = operation.ToString(),

Result = result.ToString()

};

\_context.DataInputVariants.Add(dataInputVariant);

\_context.SaveChanges();

return View("Index");

}

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

{

"ConnectionStrings": {

"DefaultConnection": "Server=mariadb; Database=9WebCalcul\_Db; Uid=root; Pwd=password; Character Set=utf8; ConvertZeroDatetime=True;"

},

"Kestrel": {

"Endpoints": {

"Http": {

"Url": "http://0.0.0.0:5009" // Заменить последние 2 цифры порта на порядковый номер из студенческого журнала. Например, порт 5012 соответствует номеру 12

}

}

},

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

},

"AllowedHosts": "\*"

}

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

{

"ConnectionStrings": {

"DefaultConnection": "Server=93.88.178.186; Port=5029; Database=9WebCalcul\_Db; Uid=root; Pwd=password; Character Set=utf8; ConvertZeroDatetime=True;"

},

"DetailedErrors": true,

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft": "Warning",

"Microsoft.Hosting.Lifetime": "Information"

}

}

}

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

using \_09\_Calculate.Data;

using Microsoft.EntityFrameworkCore;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllersWithViews();

string mariadbCS = builder.Configuration.GetConnectionString("DefaultConnection");

builder.Services.AddDbContext<CalculatorContext>(options =>

{

options.UseMySql(mariadbCS, new MySqlServerVersion(new Version(10, 5, 15)));

});

builder.Services.AddRazorPages();

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();