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»

Использование конвейера данных реального времени с Kafka

ОТЧЕТ
по практической работе № 3
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)

Студент
группы НМТ-413901 Я.В.Крашенинников

Преподаватель:
профессор, д.т.н. В.В.Лавров

Екатеринбург
2024

СОДЕРЖАНИЕ

Практическая работа

«Использование конвейера данных реального времени с Kafka»

1.1 Цель работы

Разработать модифицированную версию тестового приложения «Калькулятор» с использованием системы обмена сообщениями Apache Kafka. Реализовать структуру, обеспечивающую передачу данных между страницами ввода и получение результатов через Kafka и запись выполненных операций в базы данных MariaDB.

1.2 Ход проведения работы

Для начала установлю дополнительный пакет (Рисунок 1):

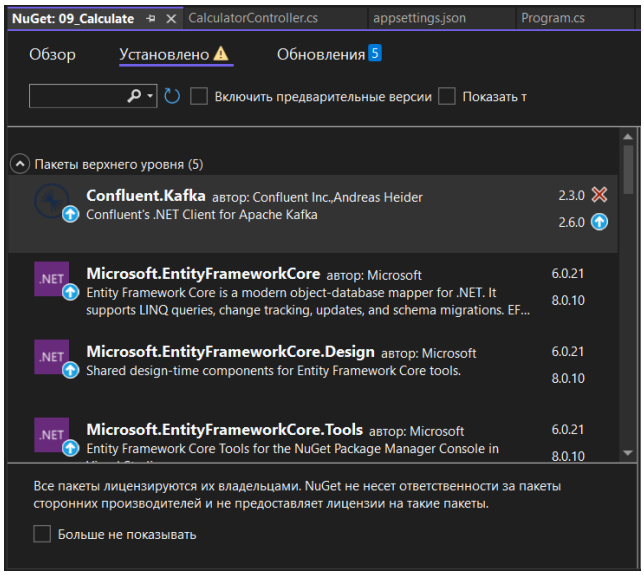

Рисунок 1 – Установка Confluent.Kafka

1.2.1 Разработка тестового приложения Visual Studio .NET Core

Добавим новую структуру с директорией Services и файлами (KafkaConsumerService.cs, KafkaProducerHandler.cs, KafkaProducerService.cs, EnumExtensions.cs, MathOperation.cs), внесём изменения в существующие файлы проекта (Рисунок 2).

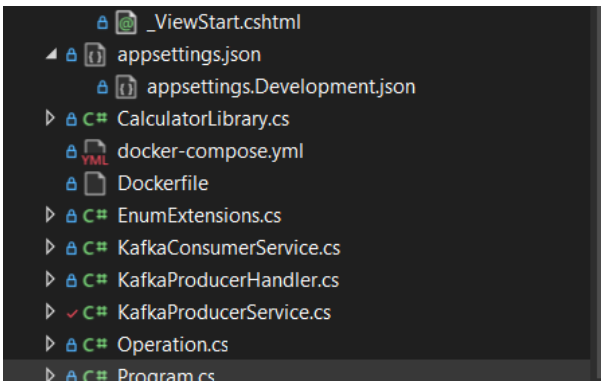

Рисунок 2 – Созданные классы

1.2.2 Размещение приложения в системе контроля версий GitHub

Протестируем работоспособность приложения и опубликуем результаты в GitHub (Рисунок 3).

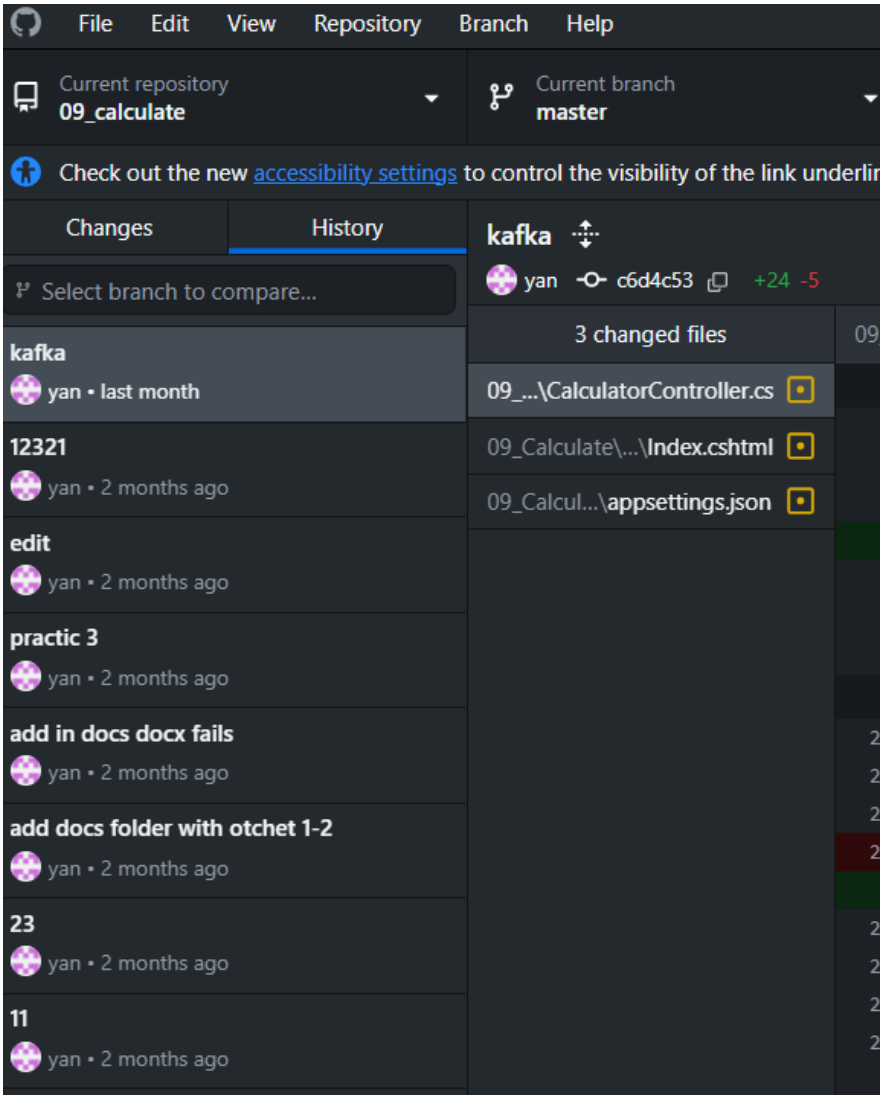

Рисунок 3 – Мои коммиты в гитхабе

1.2.3 Сборка образов

Теперь нужно клонировать репозиторий командой git clone, затем перейти в папку проекта cd, запустить контейнеры командой docker compose up -d и проверить их статус через docker ps -a.

1.2.4 Установка программы HeidiSQL для сопровождения базы данных в СУБД MariaDB

Этот шаг осуществлен в предыдущей работе. Окно программы HeidiSQL с моими настройками (Рисунок 4).

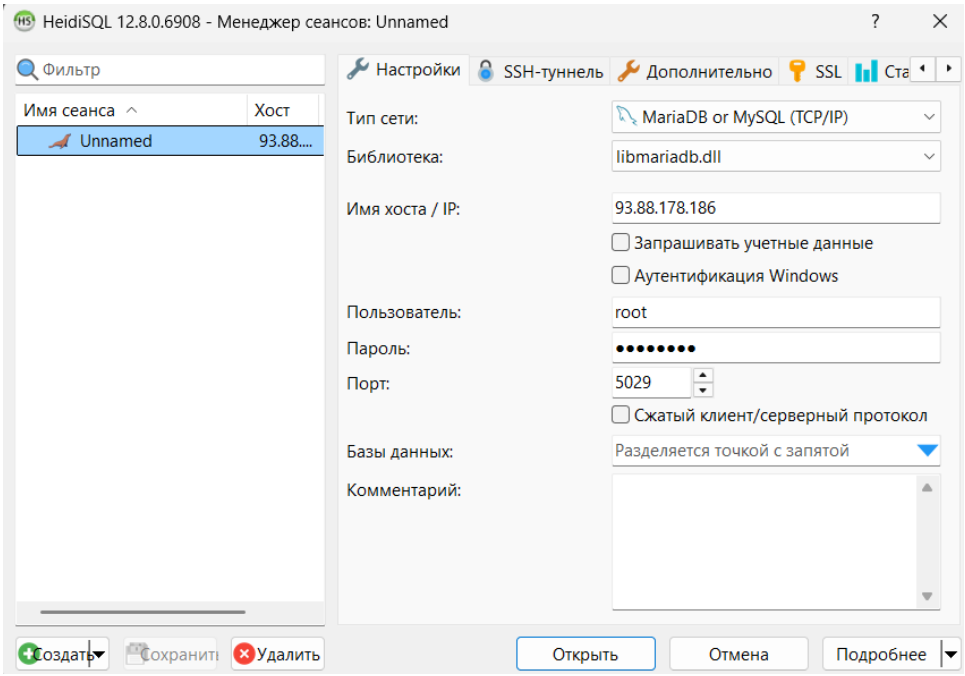

Рисунок 4 – Окно программы с БД

1.2.5 Настройка миграции базы данных в MariaDB

Данный шаг тоже был реализован в предыдущей лабораторной работе.

1.2.6 Демонстрация работы приложения брокером сообщений Kafka

Я Вам все показывал на паре ☹((произошли какие то технические шоколадки и це-лый файл со скринами куда то исчез, но Вы все лично видели и приняли! Даже разреши-ли мне сдать без основных скринов…. (Рисунок 5)

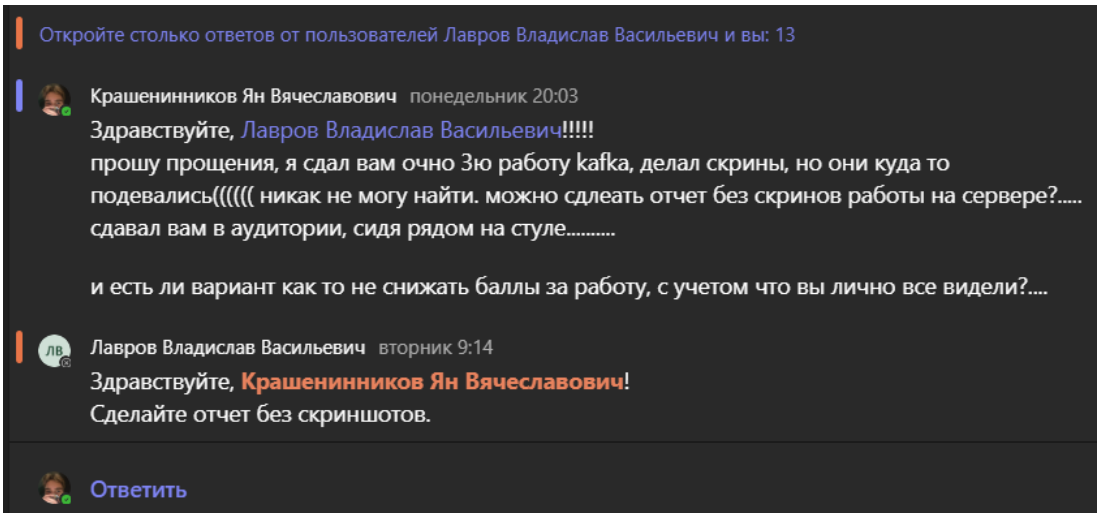

Рисунок 5 – Ваше одобрение….

1.3 Выводы

В ходе выполнения работы была разработана новая версия приложения «Калькулятор», использующая систему обмена сообщениями Apache Kafka для обработки и передачи данных между компонентами. Структура обработки позволяет реализовать обмен сообщениями между страницами ввода данных и получением результатов страниц, что обеспечивает асинхронную передачу данных и сокращение задержек. Данные о выполненных операциях передаются в свою очередь Kafka, после чего основываются на базе данных MariaDB, что обеспечивает более устойчивую и гибкую архитектуру приложения.

Использование Apache Kafka повышает масштабируемость приложений, так как позволяет эффективно распределять данные между различными

компонентами и обрабатывать запросы в режиме реального времени. Подключение пакета Confluent.Kafka расширило возможности беспроводного приложения, добавив поддержку очередных сообщений, что особенно полезно для приложений, ориентированных на поддержку и асинхронное взаимодействие.

Таким образом, изменение структуры с применением Kafka значительно повысило производительность и устойчивость приложений, что было подтверждено успешными тестами, демонстрирующими корректную работу всех компонентов. Полученные результаты позволяют утверждать, что использование Kafka – надежное решение для обработки данных в современных распределённых приложениях.

Приложение А Листинг программного кода приложения «Калькулятор» (контрол-лер)

```
using _09_Calculate.Data;
using _09_Calculate.Models;
using _09_Calculate.Services;
using Confluent.Kafka;
using Microsoft.AspNetCore.Mvc;
using System.Text.Json;
using System.Linq;
using System.Threading.Tasks;

namespace _09_Calculate.Controllers
{
   public class CalculatorController : Controller
   {
      private readonly CalculatorContext _context;
      private readonly KafkaProducerService<Null, string> _producer;

      public CalculatorController(CalculatorContext context, KafkaProducerService<Null, string> producer)
      {
         _context = context;
         _producer = producer;
      }

      /// <summary>
      /// Отображение страницы Index с данными из базы.
      /// </summary>
```

```
        public IActionResult Index()
        {
            var data = _context.DataInputVariants.OrderByDescending(x => x.ID_DataInputVariant).ToList();
            ViewBag.Data = data; // Передача данных в ViewBag
            return View(); // Возвращаем представление без передачи данных напрямую
        }

        /// <summary>
        /// Обработка запроса на вычисление.
        /// </summary>
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> ProcessingCalculationRequest(double num1, double num2, Models.Operation operation)
        {
            double result = 0;
            string errorMessage = null;

            try
            {
                switch (operation)
                {
                    case Models.Operation.Add:
                        result = num1 + num2;
                        break;
                    case Models.Operation.Subtract:
                        result = num1 - num2;
```

```
            break;
        case Models.Operation.Multiply:
            result = num1 * num2;
            break;
        case Models.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 объект для передачи в Kafka
var dataInputVariant = new DataInputVariant
{
    Operand_1 = num1,
    Operand_2 = num2,
    Type_operation = operation,
    Result = result.ToString()
```

```
            };

            // Отправка данных в Kafka
            await SendDataToKafka(dataInputVariant);

            // Сохранение данных в БД
            _context.DataInputVariants.Add(dataInputVariant);
            _context.SaveChanges();

            // Обновляем ViewBag.Data после добавления новой записи
            var updatedData = _context.DataInputVariants.OrderByDescending(x
=> x.ID_DataInputVariant).ToList();
            ViewBag.Data = updatedData;

            return View("Index");
        }

        public IActionResult Callback([FromBody] DataInputVariant
inputData)
        {
            SaveDataAndResult(inputData);
            return Ok();
        }

        /// <summary>
        /// Сохранение данных и результата в базе данных.
        /// </summary>
        private DataInputVariant SaveDataAndResult(DataInputVariant
inputData)
        {
```

```
            _context.DataInputVariants.Add(inputData);
            _context.SaveChanges();
            return inputData;
        }

        /// <summary>
        /// Отправка данных в Kafka.
        /// </summary>
        private async Task SendDataToKafka(DataInputVariant dataInputVariant)
        {
            var json = JsonSerializer.Serialize(dataInputVariant);
            await _producer.ProduceAsync("krasheninnikov", new Message<Null, string> { Value = json });
        }
    }
}
```

Приложение Б Листинг программного кода приложения «Калькулятор» (KafkaConsumerService.cs)

```
using Confluent.Kafka;
using System;
using System.Threading.Tasks;

namespace _09_Calculate.Services
{
    public class KafkaProducerService<K, V>
    {
        IProducer<K, V> kafkaHandle;

        public KafkaProducerService(KafkaProducerHandler handle)
        {
            kafkaHandle = new DependentProducerBuilder<K, V>(handle.Handle).Build();
        }

        public Task ProduceAsync(string topic, Message<K, V> message)
            => kafkaHandle.ProduceAsync(topic, message);

        public void Produce(string topic, Message<K, V> message, Action<DeliveryReport<K, V>> deliveryHandler = null)
            => kafkaHandle.Produce(topic, message, deliveryHandler);

        public void Flush(TimeSpan timeout)
            => kafkaHandle.Flush(timeout);

    }
```

}

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

```
{
 "ConnectionStrings": {
  "DefaultConnection": "Server=mariadb; Database=9WebCalcul_Db; Uid=root; Pwd=password; Character Set=utf8; ConvertZeroDatetime=True;"
 },
 "Kafka": {
  "ProducerSettings": {
   "BootstrapServers": "93.88.178.186:9094",
   "SaslMechanism": "Plain",
   "SecurityProtocol": "Plaintext"
  },
  "ConsumerSettings": {
   "BootstrapServers": "93.88.178.186:9094",
   "GroupId": "krasheninnikov",
   "SaslMechanism": "Plain",
   "SecurityProtocol": "Plaintext",
   "AutoOffsetReset": "Earliest",
   "EnableAutoCommit": true
  },
  "TopicName": "Krasheninnikov"
 },
 "Kestrel": {
  "Endpoints": {
   "Http": {
    "Url": "http://0.0.0.0:5009"
```

```
    }
   }
  },

  "Logging": {
   "LogLevel": {
    "Default": "Information",
    "Microsoft.AspNetCore": "Warning"
   }
  },
  "AllowedHosts": "*"
}
```

Приложение Г 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;"
  },
  "Kafka": {
    "ProducerSettings": {
      "BootstrapServers": "93.88.178.186:9094"
    },
    "ConsumerSettings": {
      "BootstrapServers": "93.88.178.186:9094"
    }
  },

  "DetailedErrors": true,
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "Information"
    }
  }
}
```

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

```
using _09_Calculate.Data;
using _09_Calculate.Services;
using Confluent.Kafka;
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();
builder.Services.AddHttpClient();
builder.Services.AddHostedService<KafkaConsumerService>();
builder.Services.AddSingleton<KafkaProducerHandler>();
builder.Services.AddSingleton<KafkaProducerService<Null, string>>();
```

```
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();
```