ДОДАТОК А

Перелік джерел посилання за науковими напрямами керівника та науковців кафедри програмної інженерії

11. Falatiuk H., Shirokopetleva M., Dudar Z. Investigation of Architecture and Technology Stack for e-Archive System. 2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T), Kyiv,Ukraine,8-11October2019.2019.,URL: https://doi.org/10.1109/picst47496.2019.9061407 (дата звернення: 05.04.2024).

12. DATA EXCHANGE MODEL IN THE INTERNET OF THINGS CONCEPT / I. Afanasieva et al. Telecommunications and Radio Engineering. 2019. Vol. 78,no. 10.P. 869–878. URL: https://doi.org/10.1615/telecomradeng.v78.i10.30 (date of access: 11.06.2024).

13. Аналіз результатів бенчмаркіунгу URL: https://doi.org/10.1615/telecomradeng.v78.i10.30 (date of access: 11.06.2024)

ДОДАТОК Б

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ

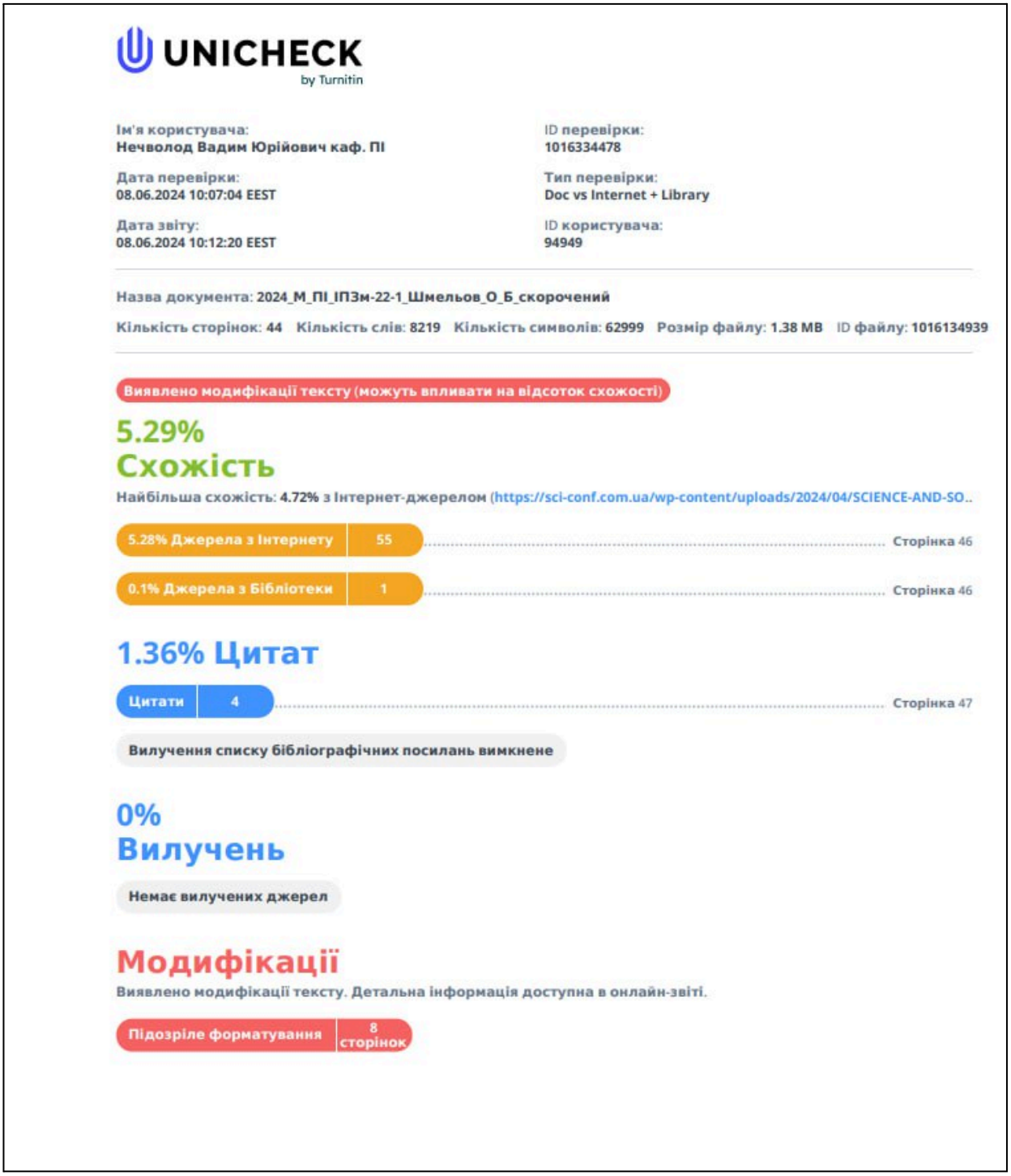


Рисунок Б.1 – Результат перевірки роботи на академічну доброчесність

ДОДАТОК В

Кодова база головного класу бенчмарку маперів

```
// See https://aka.ms/new-console-template for more information
// <auto-generated/>
using global::System;
using System.Linq.Expressions;
using System.Reflection;
using System.Runtime.CompilerServices;
using BenchmarkDotNet.Attributes;
using BenchmarkDotNet.Jobs;
using BenchmarkDotNet.Running;
using FastExpressionCompiler;
using Sigil;

public class Program
{
  public static void Main()
  {
    BenchmarkRunner.Run<Benchmark>();
    //new Benchmark().SlowReflection();
    //new Benchmark().FastReflection();
  }
}


[SimpleJob(RuntimeMoniker.Net48, baseline: true)]
[SimpleJob(RuntimeMoniker.NetCoreApp31)]
[SimpleJob(RuntimeMoniker.Net50)]
[SimpleJob(RuntimeMoniker.Net60)]
[SimpleJob(RuntimeMoniker.Net70)]
[MemoryDiagnoser()]
public class Benchmark
{
  private readonly C1 _c1;
  private readonly Func<C1, C2> _dynamicMethod, _delegate,
_builtFromExpression, _fastCompiledExpression;

  private readonly PropertyInfo setInt,
    setDecimal,
    setString;

  private readonly PropertyInfo getInt,
    getDecimal,
    getString;

  private readonly FieldInfo sourceStringField, destStringField;

  private readonly object[] args = new object[1];
  public Benchmark()
  {
    _c1 = new C1() {
      Int = 777,
      Decimal = 3.14000000000001m,
      String = "Hello",
      StringField = "World"
    };

    _dynamicMethod = CreateMethod();
```

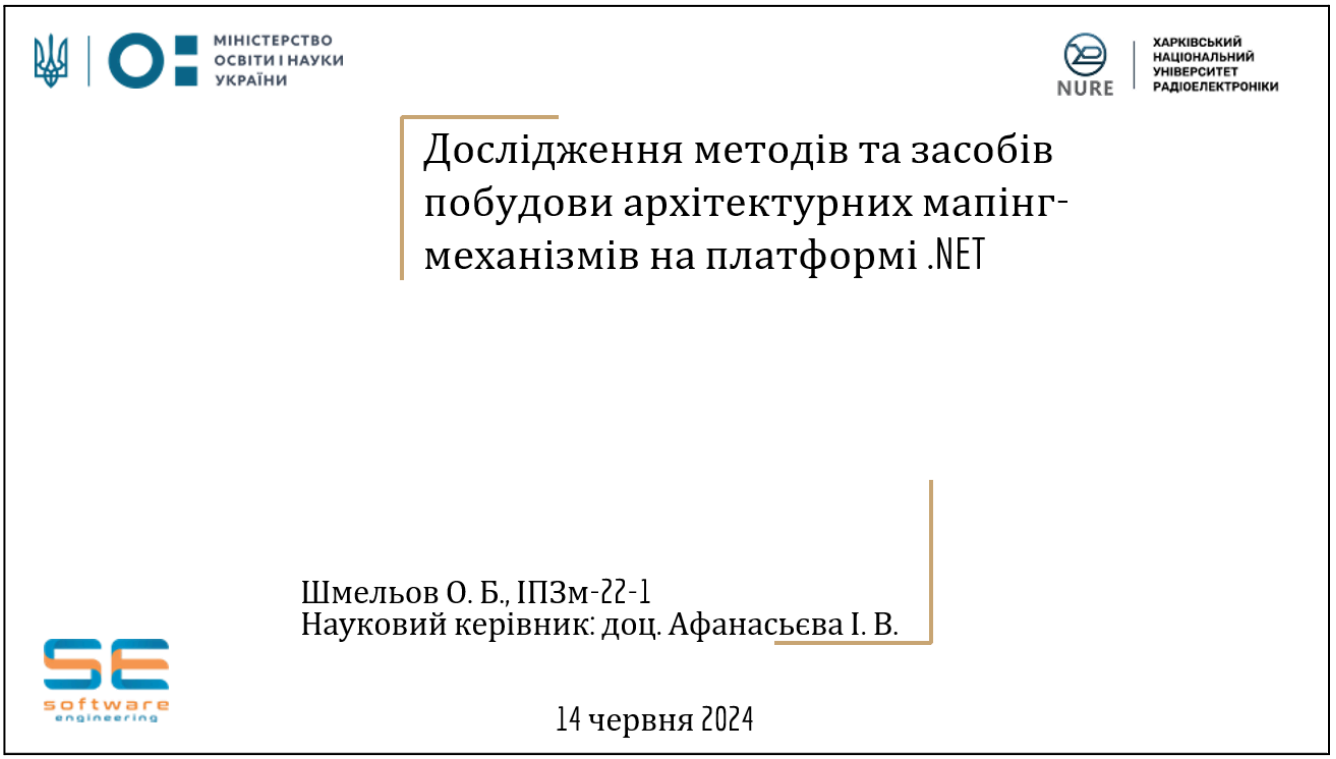
```
    _delegate = MapMethod;
    var variable = Expression.Variable(typeof(C2));
    var arg = Expression.Parameter(typeof(C1));

    Expression<Func<C1, C2>> exp = Expression.Lambda<Func<C1,
C2>>(Expression.Block(new [] { variable }, new Expression[] {
      Expression.Assign(variable, Expression.New(typeof(C2))),
      Expression.Assign(Expression.PropertyOrField(variable, "Int"),
Expression.PropertyOrField(arg, "Int")),
      Expression.Assign(Expression.PropertyOrField(variable, "Decimal"),
Expression.PropertyOrField(arg, "Decimal")),
      Expression.Assign(Expression.PropertyOrField(variable, "StringField"),
Expression.PropertyOrField(arg, "StringField")),
      Expression.Assign(Expression.PropertyOrField(variable, "String"),
Expression.PropertyOrField(arg, "String")),
      variable
    }), arg);
    _builtFromExpression = exp.Compile();
    _fastCompiledExpression = exp.CompileFast();

    var destType = typeof(C2);
    var sourceType = typeof(C1);

    getInt = sourceType.GetProperty("Int");
    getDecimal = sourceType.GetProperty("Decimal");
    getString = sourceType.GetProperty("String");

    setInt = destType.GetProperty("Int");
    setDecimal = destType.GetProperty("Decimal");
    setString = destType.GetProperty("String");

    sourceStringField = sourceType.GetField("StringField");
    destStringField = destType.GetField("StringField");
  }

  private Func<C1, C2> CreateMethod()
  {
    var emiter = Emit<Func<C1, C2>>.NewDynamicMethod("MyMethod");
    using (var local = emiter.DeclareLocal<C2>())
    {
      emiter.NewObject<C2>();
      emiter.StoreLocal(local);

      emiter.LoadLocal(local);
      emiter.LoadArgument(0);
      emiter.CallVirtual(typeof(C1).GetProperty("Int")?.GetMethod);
      emiter.CallVirtual(typeof(C2).GetProperty("Int")?.SetMethod);

      emiter.LoadLocal(local);
      emiter.LoadArgument(0);
      emiter.CallVirtual(typeof(C1).GetProperty("Decimal")?.GetMethod);
      emiter.CallVirtual(typeof(C2).GetProperty("Decimal")?.SetMethod);

      emiter.LoadLocal(local);
      emiter.LoadArgument(0);
      emiter.CallVirtual(typeof(C1).GetProperty("String")?.GetMethod);
      emiter.CallVirtual(typeof(C2).GetProperty("String")?.SetMethod);

      emiter.LoadLocal(local);
      emiter.LoadArgument(0);
      emiter.LoadField(typeof(C1).GetField("StringField"));
      emiter.StoreField(typeof(C2).GetField("StringField"));
```

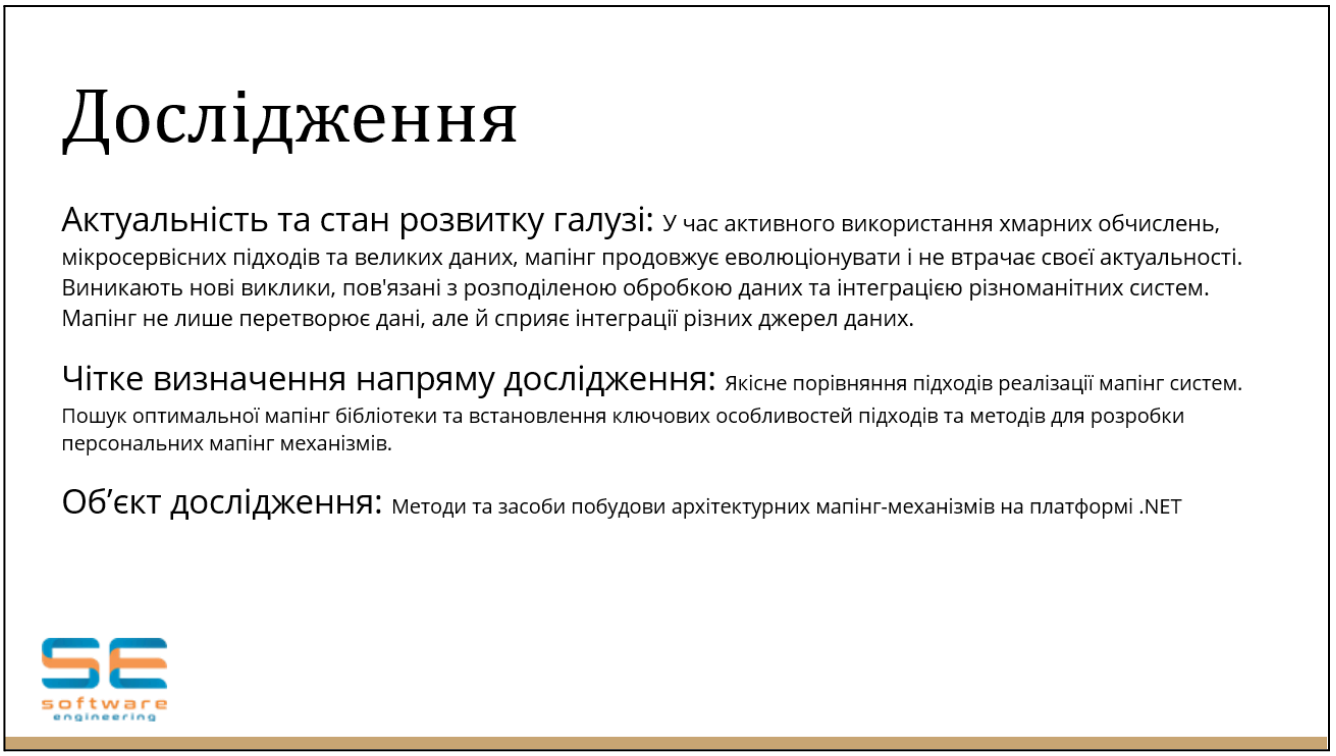
```
    emiter.LoadLocal(local);
    emiter.Return();
  }
  return emiter.CreateDelegate();
}

[MethodImpl(MethodImplOptions.NoInlining)]
private C2 MapMethod(C1 c)
{
  C2 result = new C2();
  result.Int = c.Int;
  result.Decimal = c.Decimal;
  result.String = c.String;
  result.StringField = c.StringField;

  return result;
}

[Benchmark(Baseline = true)]
public C2 Baseline() => MapMethod(_c1);

[Benchmark]
public C2 Delegate() => _delegate(_c1);

[Benchmark]
public C2 DynamicMethod() => _dynamicMethod(_c1);

[Benchmark]
public C2 CompiledExpression() => _builtFromExpression(_c1);

[Benchmark]
public C2 FastCompiledExpression() => _fastCompiledExpression(_c1);

[Benchmark]
public C2 SlowReflection()
{
  var type = _c1.GetType();
  var destType = typeof(C2);
  var c2 = Activator.CreateInstance(destType);

  foreach (var prop in type.GetProperties())
  {
    destType.GetProperty(prop.Name).SetValue(c2, prop.GetValue(_c1));
  }

  foreach (var prop in type.GetFields())
  {
    destType.GetField(prop.Name).SetValue(c2, prop.GetValue(_c1));
  }
  return (C2)c2;
}

[Benchmark]
public C2 FastReflection()
{
  var c2 = Activator.CreateInstance<C2>();

  setInt.SetValue(c2, getInt.GetValue(_c1));
  setDecimal.SetValue(c2, getDecimal.GetValue(_c1));
  setString.SetValue(c2, getString.GetValue(_c1));
  destStringField.SetValue(c2, sourceStringField.GetValue(_c1));
```

```
    return c2;
  }
}

public class C1
{
  public int Int { get; set; }
  public decimal Decimal { get; set; }
  public string StringField;
  public string String { get; set; }
}

public class C2
{
  public int Int { get; set; }
  public decimal Decimal { get; set; }
  public string StringField;
  public string String { get; set; }
}
```

ДОДАТОК Г

Слайди презентації





Огляд літератури (аналогів)

1. Аналіз предметної області: (дата звернення: 23.01.2024). Вивчення вирішуваної проблеми існуючими засобами.
2. Аналіз існуючих рішень: (дата звернення: 12.02.2024). Вивчення ефективності та наявних недоліків існуючих рішень.
3. Аналіз мапінг бібліотек: (дата звернення: 21.03.2024). Вивчення принципів роботи наявних мапінг бібліотек.
4. Аналіз source generators: (дата звернення: 10.04.2024). Детальне вивчення принципів роботи бібліотек побудованих за допомогою Source Generators
5. Аналіз reflection ammit: (дата звернення: 13.04.2024). Детальне вивчення принципів роботи бібліотек побудованих за допомогою Reflection Ammit
6. FastCompileExpression бібліотека: (дата звернення: 23.04.2024). Вивчення переваг принципів зазначеної бібліотеки за рахунок реалізації пришвидшеної компіляції Expression Trees

Постановка задачі

Чітке формулювання проблеми: У ході даної роботи буде досліджено предметну область дослідження мапінг механізмів та мапінг бібліотек, буде розібрано які є бібліотеки, далі розібрано спосіб їх роботи та їх особливості, розділено до архітектурних патернів усі архітектурні та складові елементи бібліотек.

Надалі порівняно працездатність кожного методу мапінгу відносно версії платформи для виведення протиріч та патернів. Буде розібрано результати та причини за яких такі результати сформовані.

Опис очікуваних результатів: порівняльний аналіз методів та фундаментальних підходів для створення особистих мапінг механізмів. Розроблені програмні системи-бенчмарки для аналізу продуктивності маппінг механізмів та бібліотек. Розроблено рекомендації що до використання певних методів за певних умов.

Методологія

Опис використаних методів дослідження: експериментальне порівняння бібліотек, архітектурне дослідження мапінг механізмів

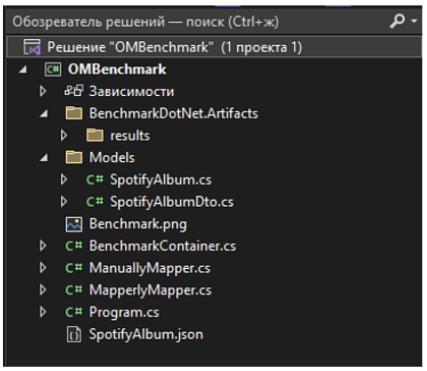
Інструментарій та технології, використані в роботі:

- Середа розробки: Visual Studio
- Платформа .NET
- Мова програмування C#
- Бібліотеки: Mapster, Automaper
- Бенчмарк-бібліотека: BenchmarkDotNet

Архітектура системи для проведення експериментального дослідження

Схема архітектури розробленої системи;

Опис ключових компонентів.



Опис програмного забезпечення, що було використано у дослідженні

Опис процесу розробки: створення бенчмарк-додатку на тестових даних для якісного порівняння досліджуваних мапінг технологій

Вибрані мови програмування та фреймворки:

- Мова програмування C#
- Середа розробки Visual Studio
- Бібліотека Automapper
- Бібліотека Mapster
- Бібліотека BenchmarkDotNet

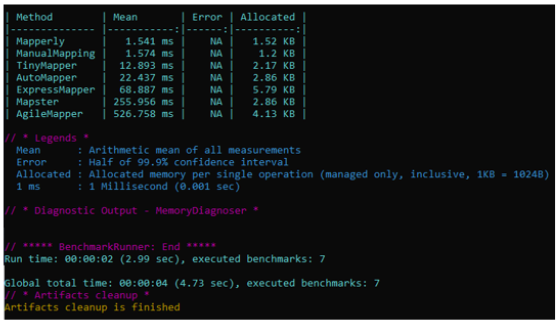
Зміст проведеного експерименту №1

Було проведено два експеременти: на тестування швидкодії бібліотек та натестування підходів до маппінгу.

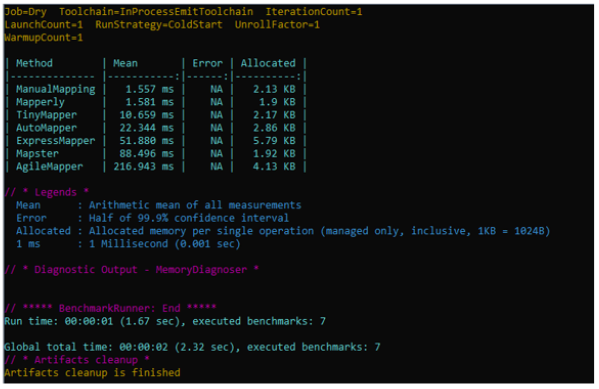
- Процес тестування включав аналіз ефективності кожної бібліотеки відносно її версії, нами було протестовано бібліотеки версії яких відрізняються функціональними можливостями версії платформі .Net для якої вона назначалась, таким чином окрім продуктивності нами будло протестовано продуктивність механізмів мапінгу для кожної версії платформи.
- Відповідно до висновків попереднього розділу – було прийнято рішення запустити програму на різних версіях .Net, а саме на 4.8 фреймворку та на LTS 6 версії.

Результати експерименту №1

Результати на 4.8 фреймворку та на LTS 6 версії.



```
| Method        | Mean       | Error | Allocated |
|-------------- |-----------:|------:|----------:|
| Mapperly      |   1.541 ms |    NA |   1.52 KB |
| ManualMapping |   1.574 ms |    NA |   1.2 KB  |
| TinyMapper    |  12.893 ms |    NA |   2.17 KB |
| AutoMapper    |  22.437 ms |    NA |   2.86 KB |
| ExpressMapper |  68.887 ms |    NA |   5.79 KB |
| Mapster       | 255.956 ms |    NA |   2.86 KB |
| AgileMapper   | 526.758 ms |    NA |   4.13 KB |

// * Legends *
  Mean      : Arithmetic mean of all measurements
  Error     : Half of 99.9% confidence interval
  Allocated : Allocated memory per single operation (managed only, inclusive, 1KB = 1024B)
  1 ms      : 1 Millisecond (0.001 sec)

// * Diagnostic Output - MemoryDiagnoser *


// ***** BenchmarkRunner: End *****
Run time: 00:00:02 (2.99 sec), executed benchmarks: 7

Global total time: 00:00:04 (4.73 sec), executed benchmarks: 7
// * Artifacts cleanup *
Artifacts cleanup is finished
```



```
Job-DRY  Toolchain=InProcessEmitToolchain  IterationCount=1
LaunchCount=1  RunStrategy=ColdStart  UnrollFactor=1
WarmupCount=1

| Method        | Mean       | Error | Allocated |
|-------------- |-----------:|------:|----------:|
| ManualMapping |   1.557 ms |    NA |   2.13 KB |
| Mapperly      |   1.581 ms |    NA |    1.9 KB |
| TinyMapper    |  10.659 ms |    NA |   2.17 KB |
| AutoMapper    |  22.344 ms |    NA |   2.86 KB |
| ExpressMapper |  51.880 ms |    NA |   5.79 KB |
| Mapster       |  88.496 ms |    NA |   1.92 KB |
| AgileMapper   | 216.943 ms |    NA |   4.13 KB |

// * Legends *
  Mean      : Arithmetic mean of all measurements
  Error     : Half of 99.9% confidence interval
  Allocated : Allocated memory per single operation (managed only, inclusive, 1KB = 1024B)
  1 ms      : 1 Millisecond (0.001 sec)

// * Diagnostic Output - MemoryDiagnoser *


// ***** BenchmarkRunner: End *****
Run time: 00:00:01 (1.67 sec), executed benchmarks: 7

Global total time: 00:00:02 (2.32 sec), executed benchmarks: 7
// * Artifacts cleanup *
Artifacts cleanup is finished
```

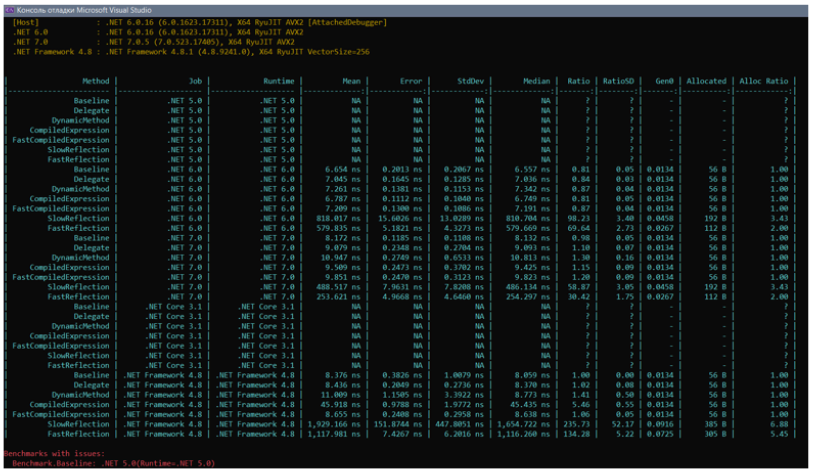
Зміст проведеного експерименту №2

Тестування підходів до маппінгу.
У процесі підготовки до тесту нами було реалізовано:

- Розроблено тестові моделі;
- Використано атрибути BenchmarkDotNet;
- Встановлено метрики;
- Додано та розглянуто методи Delegate та Baseline;
- Проаналізовано результати.

Результати експерименту №2

Результат відпрацювання бенчмарку та впорядковані результати:



```
Консоль отладки Microsoft Visual Studio
[Host]           : .NET 6.0.16 (6.0.1623.17311), X64 RyuJIT AVX2 [AttachedDebugger]
.NET 6.0         : .NET 6.0.16 (6.0.1623.17311), X64 RyuJIT AVX2
.NET 7.0         : .NET 7.0.5 (7.0.523.17405), X64 RyuJIT AVX2
.NET Framework 4.8 : .NET Framework 4.8.1 (4.8.9241.0), X64 RyuJIT VectorSize=256

|              Method |              Job |          Runtime |         Mean |       Error |      StdDev |       Median | Ratio | RatioSD |  Gen0 | Allocated | Alloc Ratio |
|-------------------- |----------------- |----------------- |------------- |------------ |------------ |------------- |------ |-------- |------ |---------- |------------ |
|            Baseline |          .NET 5.0 |          .NET 5.0 |           NA |          NA |          NA |           NA |     ? |       ? |     - |         - |           ? |
|            Delegate |          .NET 5.0 |          .NET 5.0 |           NA |          NA |          NA |           NA |     ? |       ? |     - |         - |           ? |
|         DynamicMethod |          .NET 5.0 |          .NET 5.0 |           NA |          NA |          NA |           NA |     ? |       ? |     - |         - |           ? |
|     CompiledExpression |          .NET 5.0 |          .NET 5.0 |           NA |          NA |          NA |           NA |     ? |       ? |     - |         - |           ? |
| FastCompiledExpression |          .NET 5.0 |          .NET 5.0 |           NA |          NA |          NA |           NA |     ? |       ? |     - |         - |           ? |
|         SlowReflection |          .NET 5.0 |          .NET 5.0 |           NA |          NA |          NA |           NA |     ? |       ? |     - |         - |           ? |
|         FastReflection |          .NET 5.0 |          .NET 5.0 |           NA |          NA |          NA |           NA |     ? |       ? |     - |         - |           ? |
|            Baseline |          .NET 6.0 |          .NET 6.0 |     6.654 ns |   0.2013 ns |   0.2067 ns |     6.557 ns |  0.81 |    0.05 |  0.0134 |      56 B |        1.00 |
|            Delegate |          .NET 6.0 |          .NET 6.0 |     7.045 ns |   0.1645 ns |   0.1285 ns |     7.036 ns |  0.84 |    0.03 |  0.0134 |      56 B |        1.00 |
|         DynamicMethod |          .NET 6.0 |          .NET 6.0 |     7.261 ns |   0.1381 ns |   0.1153 ns |     7.342 ns |  0.87 |    0.04 |  0.0134 |      56 B |        1.00 |
|     CompiledExpression |          .NET 6.0 |          .NET 6.0 |     6.787 ns |   0.1112 ns |   0.1040 ns |     6.749 ns |  0.81 |    0.05 |  0.0134 |      56 B |        1.00 |
| FastCompiledExpression |          .NET 6.0 |          .NET 6.0 |     7.209 ns |   0.1300 ns |   0.1086 ns |     7.191 ns |  0.87 |    0.04 |  0.0134 |      56 B |        1.00 |
|         SlowReflection |          .NET 6.0 |          .NET 6.0 |   818.017 ns |  15.6026 ns |  13.0289 ns |   810.704 ns | 98.23 |    3.40 |  0.0458 |     192 B |        3.43 |
|         FastReflection |          .NET 6.0 |          .NET 6.0 |   579.835 ns |   5.1821 ns |   4.3273 ns |   579.669 ns | 69.64 |    2.73 |  0.0267 |     112 B |        2.00 |
|            Baseline |          .NET 7.0 |          .NET 7.0 |     8.172 ns |   0.1185 ns |   0.1108 ns |     8.132 ns |  0.98 |    0.05 |  0.0134 |      56 B |        1.00 |
|            Delegate |          .NET 7.0 |          .NET 7.0 |     9.079 ns |   0.2348 ns |   0.2704 ns |     9.093 ns |  1.10 |    0.07 |  0.0134 |      56 B |        1.00 |
|         DynamicMethod |          .NET 7.0 |          .NET 7.0 |    10.947 ns |   0.2749 ns |   0.6533 ns |    10.813 ns |  1.30 |    0.16 |  0.0134 |      56 B |        1.00 |
|     CompiledExpression |          .NET 7.0 |          .NET 7.0 |     9.509 ns |   0.2473 ns |   0.3702 ns |     9.425 ns |  1.15 |    0.09 |  0.0134 |      56 B |        1.00 |
| FastCompiledExpression |          .NET 7.0 |          .NET 7.0 |     9.851 ns |   0.2470 ns |   0.3123 ns |     9.823 ns |  1.20 |    0.09 |  0.0134 |      56 B |        1.00 |
|         SlowReflection |          .NET 7.0 |          .NET 7.0 |   488.517 ns |   7.9631 ns |   7.8208 ns |   486.134 ns | 58.87 |    3.05 |  0.0458 |     192 B |        3.43 |
|         FastReflection |          .NET 7.0 |          .NET 7.0 |   253.621 ns |   4.9668 ns |   4.6460 ns |   254.297 ns | 30.42 |    1.75 |  0.0267 |     112 B |        2.00 |
|            Baseline |      .NET Core 3.1 |      .NET Core 3.1 |           NA |          NA |          NA |           NA |     ? |       ? |     - |         - |           ? |
|            Delegate |      .NET Core 3.1 |      .NET Core 3.1 |           NA |          NA |          NA |           NA |     ? |       ? |     - |         - |           ? |
|         DynamicMethod |      .NET Core 3.1 |      .NET Core 3.1 |           NA |          NA |          NA |           NA |     ? |       ? |     - |         - |           ? |
|     CompiledExpression |      .NET Core 3.1 |      .NET Core 3.1 |           NA |          NA |          NA |           NA |     ? |       ? |     - |         - |           ? |
| FastCompiledExpression |      .NET Core 3.1 |      .NET Core 3.1 |           NA |          NA |          NA |           NA |     ? |       ? |     - |         - |           ? |
|         SlowReflection |      .NET Core 3.1 |      .NET Core 3.1 |           NA |          NA |          NA |           NA |     ? |       ? |     - |         - |           ? |
|         FastReflection |      .NET Core 3.1 |      .NET Core 3.1 |           NA |          NA |          NA |           NA |     ? |       ? |     - |         - |           ? |
|            Baseline | .NET Framework 4.8 | .NET Framework 4.8 |     8.376 ns |   0.3826 ns |   1.0079 ns |     8.059 ns |  1.00 |    0.00 |  0.0134 |      56 B |        1.00 |
|            Delegate | .NET Framework 4.8 | .NET Framework 4.8 |     8.436 ns |   0.2049 ns |   0.2736 ns |     8.370 ns |  1.02 |    0.08 |  0.0134 |      56 B |        1.00 |
|         DynamicMethod | .NET Framework 4.8 | .NET Framework 4.8 |    11.009 ns |   1.1505 ns |   3.3922 ns |     8.773 ns |  1.41 |    0.58 |  0.0134 |      56 B |        1.00 |
|     CompiledExpression | .NET Framework 4.8 | .NET Framework 4.8 |    45.918 ns |   0.9788 ns |   1.9772 ns |    45.435 ns |  5.46 |    0.55 |  0.0134 |      56 B |        1.00 |
| FastCompiledExpression | .NET Framework 4.8 | .NET Framework 4.8 |     8.655 ns |   0.2408 ns |   0.2958 ns |     8.638 ns |  1.06 |    0.05 |  0.0134 |      56 B |        1.00 |
|         SlowReflection | .NET Framework 4.8 | .NET Framework 4.8 | 1,929.166 ns | 151.8744 ns | 447.8051 ns | 1,654.722 ns | 235.73 |   52.17 |  0.0916 |     385 B |        6.88 |
|         FastReflection | .NET Framework 4.8 | .NET Framework 4.8 | 1,117.981 ns |   7.4267 ns |   6.2016 ns | 1,116.260 ns | 134.28 |    5.22 |  0.0725 |     305 B |        5.45 |

Benchmarks with issues:
  Benchmark.Baseline: .NET 5.0(Runtime=.NET 5.0)
```

Результати експерименту №2

Результат відпрацювання бенчмарку та впорядковані результати:

Method	Job	Mean	Error	StdDev	Median	Ratio	RatioSD	Gen0	Allocated	Alloc Ratio
Baseline	Framework 4.8	7.794 ns	0.4898 ns	1.3571 ns	7.496 ns	1.00	0.00	0.0089	56 B	1.00
Baseline	.NET Core 3.1	8.708 ns	0.2633 ns	0.7682 ns	8.588 ns	1.16	0.22	0.0089	56 B	1.00
Baseline	.NET 5.0	6.864 ns	0.1778 ns	0.4775 ns	6.803 ns	0.80	0.14	0.0089	56 B	1.00
Baseline	.NET 6.0	7.159 ns	0.2613 ns	0.7622 ns	7.019 ns	0.95	0.18	0.0089	56 B	1.00
Baseline	.NET 7.0	9.319 ns	0.3673 ns	1.0655 ns	9.246 ns	1.23	0.25	0.0089	56 B	1.00
Delegate	Framework 4.8	7.974 ns	0.2216 ns	0.4910 ns	7.948 ns	1.09	0.19	0.0089	56 B	1.00
Delegate	.NET Core 3.1	8.180 ns	0.2269 ns	0.6736 ns	8.101 ns	1.12	0.18	0.0089	56 B	1.00
Delegate	.NET 5.0	8.224 ns	0.2436 ns	0.7028 ns	8.164 ns	1.08	0.19	0.0089	56 B	1.00
Delegate	.NET 6.0	9.212 ns	0.3180 ns	0.8978 ns	9.139 ns	1.22	0.24	0.0089	56 B	1.00
Delegate	.NET 7.0	10.854 ns	0.5694 ns	1.6611 ns	10.593 ns	1.43	0.31	0.0089	56 B	1.00
SlowReflection	Framework 4.8	1,610.503 ns	32.0085 ns	63.9245 ns	1,611.465 ns	224.01	32.66	0.0610	385 B	6.88
SlowReflection	.NET Core 3.1	1,396.654 ns	94.5429 ns	278.7618 ns	1,250.371 ns	189.04	52.35	0.0610	384 B	6.86
SlowReflection	.NET 5.0	1,080.053 ns	21.4425 ns	47.9593 ns	1,078.085 ns	146.86	21.96	0.0610	384 B	6.86
SlowReflection	.NET 6.0	790.373 ns	15.8412 ns	22.7190 ns	786.504 ns	104.22	18.81	0.0305	192 B	3.43
SlowReflection	.NET 7.0	439.142 ns	8.7813 ns	15.1470 ns	436.838 ns	59.86	9.07	0.0305	192 B	3.43
FastReflection	Framework 4.8	1,051.206 ns	20.8693 ns	44.0284 ns	1,042.916 ns	144.90	21.24	0.0477	305 B	5.45
FastReflection	.NET Core 3.1	846.239 ns	16.9672 ns	38.9849 ns	846.052 ns	114.68	20.01	0.0477	304 B	5.43
FastReflection	.NET 5.0	735.505 ns	14.2707 ns	21.3598 ns	729.531 ns	97.61	16.96	0.0477	304 B	5.43
FastReflection	.NET 6.0	868.921 ns	125.6284 ns	370.4181 ns	594.881 ns	103.01	44.64	0.0172	112 B	2.00
FastReflection	.NET 7.0	250.846 ns	4.8429 ns	11.6961 ns	250.167 ns	33.11	6.15	0.0176	112 B	2.00
CompiledExpression	Framework 4.8	39.070 ns	0.8423 ns	1.7816 ns	39.063 ns	5.45	0.88	0.0089	56 B	1.00
CompiledExpression	.NET Core 3.1	9.232 ns	0.5360 ns	1.5805 ns	9.259 ns	1.21	0.25	0.0089	56 B	1.00
CompiledExpression	.NET 5.0	7.738 ns	0.2609 ns	0.7610 ns	7.791 ns	1.02	0.20	0.0089	56 B	1.00
CompiledExpression	.NET 6.0	8.006 ns	0.2431 ns	0.7129 ns	7.881 ns	1.06	0.19	0.0089	56 B	1.00
CompiledExpression	.NET 7.0	8.408 ns	0.2392 ns	0.6902 ns	8.338 ns	1.10	0.18	0.0089	56 B	1.00
FastCompiledExpression	Framework 4.8	7.705 ns	0.2189 ns	0.4520 ns	7.738 ns	1.07	0.18	0.0089	56 B	1.00
FastCompiledExpression	.NET Core 3.1	9.285 ns	0.3705 ns	1.0924 ns	9.443 ns	1.21	0.20	0.0089	56 B	1.00
FastCompiledExpression	.NET 5.0	8.002 ns	0.3092 ns	0.8969 ns	7.934 ns	1.05	0.16	0.0089	56 B	1.00
FastCompiledExpression	.NET 6.0	7.467 ns	0.2026 ns	0.4138 ns	7.398 ns	1.04	0.16	0.0089	56 B	1.00
FastCompiledExpression	.NET 7.0	9.014 ns	0.2325 ns	0.5526 ns	8.988 ns	1.20	0.21	0.0089	56 B	1.00
DynamicMethod	Framework 4.8	8.103 ns	0.2260 ns	0.4768 ns	8.054 ns	1.12	0.17	0.0089	56 B	1.00
DynamicMethod	.NET Core 3.1	8.193 ns	0.2266 ns	0.5600 ns	8.254 ns	1.08	0.21	0.0089	56 B	1.00
DynamicMethod	.NET 5.0	6.892 ns	0.1918 ns	0.3098 ns	6.831 ns	0.93	0.15	0.0089	56 B	1.00
DynamicMethod	.NET 6.0	7.975 ns	0.2190 ns	0.2998 ns	7.886 ns	1.04	0.20	0.0089	56 B	1.00
DynamicMethod	.NET 7.0	10.247 ns	0.3229 ns	0.9367 ns	10.045 ns	1.36	0.23	0.0089	56 B	1.00

Аналіз отриманих результатів

- За результатами ми бачимо деякі важливі та неочевидні патерни, такі як: наш динамічний метод – повільніший за expression, fust expression;
- fust expression api – яка повинна бути швидшою за звичайний expressionпрацює повільніше, хоча вони заявляють що швидше.
- Ми бачимо причину появи FustCompile – з'явилась вона тому що на Net4.8 звичайна компіляція працювала дуже повільно, тож якщо використовувати цю варсію – варто використовувати відповідно FustCompile.
- Також варто зазначити що хоча FustCompile бібліотека і не працює швидше компіляції – але вона працює ефективніше на початковому етапі компіляції, у «холодному старті»

Висновки

- Нами було виявлено найефективнішу бібліотеку для мапінгу, проранжовано відносно версії платформи. Це було зроблено за допомогою бенчмарку реалізованого нами із використанням BenchmarkDotNet.
- Нами було проведено порівняння різних методів маппінгу за допомогою бенчмарк модулю в який ми імпліментували різні методи та різні елементи збору метрик, для наших методів: BaseLine, DynamicMethod, Delegate, CompiledExpression, FastCompiledExpression, SlowReflection, та FastReflection. Це допомогло встановити, який з методів найбільш підходить для конкретних сценаріїв використання.
- Це дослідження допоможе розробникам прийняти обґрунтоване рішення про використання тієї чи іншої бібліотеки або підходу до маппінгу, виходячи з потреб їхнього проекту або версії платформи.

Публікація результатів

сертифікат про участь у конференціях:

копія опублікованої статті: Посилання

Підсумки

Результати нашого дослідження будуть актуальні та необхідні розробникам які імплементують той чи інший мапер у систему на певній версії платформи, або розробляють свій, також дослідження допоможе в виборі бібліотеки для мапінгу.

Надалі дослідження може вдосконалювати свої дані до нових версій платформи або бібліотек мапінгу.

ДОДАТОК Д

Апробація результатів роботи

УДК 004.6

ДОСЛІДЖЕННЯ МЕТОДІВ ТА АЛГОРИТМІВ РОБОТИ З ПОВНОТЕКСТОВИМИ ІНДЕКСАМИ

Шмельов О.Б.

Науковий керівник – доц. Афанасьєва І.В.

Харківський національний університет радіоелектроніки, каф. ПІ

М. Харків, Україна

e-mail: oleh.shmelov@nure.ua

In accordance with the dynamic development of technologies and growing volumes of data, information systems are continuously developing, introducing new approaches and methodologies for effective work with data. The .NET platform, which offers a wide range of opportunities for modern software development, deserves special attention in this context.

Data mapping is becoming an important part of many projects, enabling the efficient integration, transformation and management of information between different parts of the system. This includes moving data between different application layers, converting between object models and relational databases, and processing data to ensure compliance with business logic.

Актуальність вибору маппінг бібліотеки та методу для створення особистого мапера на платформі .NET обумовлена кількома ключовими факторами, які впливають на ефективність, швидкість та якість розробки програмного забезпечення.

Обробка даних є фундаментальною частиною більшості сучасних програм, і вибір ефективного механізму маппінгу безпосередньо впливає на продуктивність програми. Неправильний вибір може призвести до зайвого навантаження на систему та зниження продуктивності.

У великих та складних системах необхідна висока гнучкість маппінгу, щоб адаптуватися до змін у бізнес логіки та структурі даних. Вибір методу маппінгу та бібліотеки повинен бути здатний задовольнити потреби масштабування та розвитку проекту.

Вибір між використанням готових бібліотек, таких як AutoMapper або Mapster, і створенням власного мапера за допомогою методів, як Source Generators або Reflection Emit, також впливає на складність коду, швидкість розробки та легкість підтримки програми[3].

Цей метод забезпечує високу продуктивність, оскільки код генерується на етапі компіляції, знижуючи навантаження в ран таймі. Він також підходить для сценаріїв, де потрібно чітке визначення типів та сильна типізація.

Хоча цей метод може бути більш гнучким, оскільки дозволяє динамічно створювати код в рантаймі, він може бути менш продуктивним у порівнянні з Source Generators. Reflection Emit вимагає глибшого розуміння внутрішньої роботи .NET та може бути складнішим у підтримці.

У виборі підходу до маппінгу також важливо враховувати сумісність з іншими технологіями та фреймворками, використовуваними в проекті, такими як ASP.NET, Entity Framework, або мікросервісна архітектура.

Враховуючи ці фактори, вибір між використанням готових маппінг бібліотек маппінгу або створенням власного мапера за допомогою Source Generators чи Reflection Emit стає особливо актуальним.

Цей вибір має стратегічне значення, адже від нього залежить здатність проекту ефективно адаптуватися до змін, швидкість розробки, а також загальна продуктивність та підтримуваність системи.

У ході даної роботи нами буде досліджено предметну область дослідження маппінг механізмів та маппінг бібліотек, ми розберемо які є бібліотеки, далі розберемо як вони працюють та їх особливості, розділимо до архітектурних патернів усі архітектурні та складові елементи бібліотек.

Ми проведемо порівняльний аналіз методів та фундаментальних підходів для створення особистих маппінг механізмів.

Ця робота спрямована на виявлення критеріїв для вибору способу маппінгу об’єктів на певному проекті.

Реалізація наукового дослідження складається з наступних етапів:

- аналіз предметної області;
- аналіз архітектурних особливостей сучасних маппінг бібліотек;

- знаходження фундаментальних закономірностей у мапінг механізмах;
- порівняльний аналіз та модифікація існуючих рішень.

Після детального аналізу підходів, а саме – проаналізувавши дві бібліотеки для мапінгу mapster vs automaper, можна зробити загальний висновок що mapster – це повноцінна бібліотека, яка має функціональність, що можна порівняти з AutoMapper[1].

У багатьох сценаріях Mapster простіше в налаштуванні через те, що немає необхідності явно задавати вихідний та цільовий типи для моделей доти, доки не потрібні додаткові налаштування мапінгу властивостей.

По продуктивності Mapster перевершує AutoMapper, це можна побачити на рисунку 1, де наведено результати бенчмарку, де видно що Mapster перевершує AutoMapper за швидкістю роботи у всіх сценаріях, більшості сценаріїв на 30-50%, а деяких сценаріях більше, ніж у 2 разу. При цьому споживання пам'яті або не змінюється або зменшується.

Method	Mean	Error	StdDev	Ratio	Allocated
AutoMapper_SimpleMapping	304.29 us	0.855 us	11.594 us	1.290	109.38 KB
Mapster_SimpleMapping	235.87 us	0.628 us	8.492 us	1.000	109.38 KB
AutoMapper_ListMapping	208.90 us	0.234 us	2.949 us	1.021	125.59 KB
Mapster_ListMapping	204.54 us	0.357 us	4.846 us	1.000	117.24 KB
AutoMapper_NestedMapping	128.83 us	0.285 us	6.069 us	2.079	117.19 KB
Mapster_NestedMapping	61.94 us	0.262 us	2.914 us	1.000	117.19 KB
AutoMapper_FlattenedMapping	88.15 us	0.197 us	2.666 us	2.862	23.44 KB
Mapster_FlattenedMapping	30.79 us	0.054 us	0.735 us	1.000	23.44 KB
AutoMapper_CustomPropertyMapping	175.77 us	0.517 us	6.859 us	1.551	73.96 KB
Mapster_CustomPropertyMapping	113.29 us	0.291 us	3.951 us	1.000	73.78 KB
AutoMapper_ReverseMapping	118.53 us	0.244 us	3.495 us	2.576	46.88 KB
Mapster_ReverseMapping	46.00 us	0.140 us	1.595 us	1.000	46.88 KB
AutoMapper_AttributeMapping	301.95 us	0.698 us	9.207 us	1.296	85.94 KB
Mapster_AttributeMapping	232.81 us	0.543 us	7.349 us	1.000	85.94 KB

Рисунок 1 – Рузультати проведення бенчмарку для mapster vs. Automaper.

У всіх сценаріях, а деяких дає і виграш у кількості споживаної пам'яті. Бібліотека існує вже 7 років, має багато користувачів та продовжує розвиватися. У різних бенчмарках Mapster займає перше місце за продуктивністю серед бібліотек object-object мапінгу.

Далі планується порівняння різних методів маппінгу: BaseLine, DynamicMethod, Delegate, CompiledExpression, FastCompiledExpression, SlowReflection, та FastReflection. Це допоможе встановити, який з методів найбільш підходить для конкретних сценаріїв використання.

ДОДАТОК Е

Експертний висновок результатів перевірки кваліфікаційної роботи на відповідність оформлення вимогам ДСТУ 3008:2015

Експертний висновок результатів перевірки кваліфікаційної роботи

студент (посада) | програмної інженерії (кафедра) | ІПЗм-22-1 (група)

Шмельов О. Б.
(прізвище, ім'я, по батькові)

Зауваження

Пункт ДСТУ 3008-2015	Зміст пункту	Сторінка кваліфікаційної роботи
1	2	3
	7.1 Загальні положення	
	7.3 Нумерація сторінок звіту	
	7.4 Нумерація розділів, підрозділів, пунктів, підпунктів	
	7.5 Рисунки	
	7.6 Таблиці	
7.6.4	Таблиці нумерують наскрізно арабськими цифрами, крім таблиць у додатках. Дозволено таблиці нумерувати в межах розділу. У цьому разі номер таблиці складається з номера розділу та порядкового номера таблиці, відокремлених крапкою, наприклад, «Таблиця 2.1» — перша таблиця другого розділу.	51
7.6.9	Якщо рядки або колонки таблиці виходять за межі формату сторінки, таблицю поділяють на частини, розміщуючи одну частину під іншою або поруч, чи переносять частину таблиці на наступну сторінку. У кожній частині таблиці повторюють її головку та боковик. У разі поділу таблиці на частини дозволено її головку чи боковик заміняти відповідно номерами колонок або рядків, нумеруючи їх арабськими цифрами в першій частині таблиці. Слово «Таблиця » подають лише один раз над першою частиною таблиці. Над іншими частинами таблиці з абзацного відступу друкують «Продовження таблиці » або «Кінець таблиці ____ » без повторення її назви.	36
	7.7 Переліки	
	7.8 Примітки	
	7.9 Виноски	
	7.10 Формули та рівняння	
	7.11 Посилання	
	7.13 Список авторів	
	7.14 Скорочення та умовні позначки	
	7.15 Додатки	

Рисунок Е.1 - Експертний висновок результатів перевірки кваліфікаційної роботи на відповідність оформлення вимогам ДСТУ 3008:2015

Методичні вказівки до виконання кваліфікаційної роботи магістра… **ЗАТВЕРДЖЕНО** кафедрою ІІІ протокол № 5 від 13.11.2023р. 3.2 Оформлення пояснювальної записки згідно з ДСТУ 3008:2015 Звіти у сфері науки і техніки. Структура та правила оформлювання. **Шаблон** затверджений засіданням кафедри №3 від 16.10.2023.	Рисунок повинен розміщуватися одразу після його згадування у тексті, або на наступній сторінці. **Під рисунком повинен бути підпис із словом Рисунок, порядковим номером цього рисунку, через тире з великої літери – назва рисунку та в круглих дужках вказується джерело з якого взятий цей рисунок, або то, що його виконано самостійно.**	18, далі за текстом.
Методичні вказівки до виконання кваліфікаційної роботи магістра… **ЗАТВЕРДЖЕНО** кафедрою ІІІ протокол № 5 від 13.11.2023р. 3.2 Оформлення пояснювальної записки згідно з ДСТУ 3008:2015 Звіти у сфері науки і техніки. Структура та правила оформлювання. **Шаблон** затверджений засіданням кафедри №3 від 16.10.2023.	Назву таблиці друкують з великої літери і розміщують над таблицею з абзацного відступу та в круглих дужках вказується джерело з якого взята ця таблиця. ПРИКЛАД: шаблон, стор.15	35, далі за текстом.
Методичні вказівки до виконання кваліфікаційної роботи магістра… **ЗАТВЕРДЖЕНО** кафедрою ІІІ протокол № 5 від 13.11.2023р. 3.2 Оформлення пояснювальної записки згідно з ДСТУ 3008:2015 Звіти у сфері науки і техніки. Структура та правила оформлювання. **Шаблон** затверджений засіданням кафедри №3 від 16.10.2023.	Додатки нумеруються за допомогою літер української абетки. Слово ДОДАТОК та його назва розташовуються посередині сторінки без абзацного відступу, пишеться заголовними літерами звичайним начертанням. Після заголовку ставиться один пустий рядок	56, далі за текстом.
Методичні вказівки до виконання кваліфікаційної роботи магістра… **ЗАТВЕРДЖЕНО** кафедрою ІІІ протокол № 5 від 13.11.2023р. 3.2 Оформлення пояснювальної записки згідно з ДСТУ 3008:2015 Звіти у сфері науки і техніки. Структура та правила оформлювання. **Шаблон** затверджений засіданням кафедри №3 від 16.10.2023.	Приклад оформлення титульного листа: **Шаблон** затверджений засіданням кафедри №3 від 16.10.2023 стор.1	1

Рисунок Е.2 - Експертний висновок результатів перевірки кваліфікаційної роботи на відповідність оформлення вимогам ДСТУ 3008:2015

Методичні вказівки до виконання кваліфікаційної роботи магістра… **ЗАТВЕРДЖЕНО** кафедрою ПІ протокол № 5 від 13.11.2023р. 3.2 Оформлення пояснювальної записки згідно з ДСТУ 3008:2015 Звіти у сфері науки і техніки. Структура та правила оформлювання. **Шаблон** затверджений засіданням кафедри №3 від 16.10.2023.	Не допускається розміщення підзаголовку на одній сторінці, а тексту підрозділу на іншій. **Шаблон затверджений засіданням кафедри №3 від 16.10.2023. стор.12**	48
Методичні вказівки до виконання кваліфікаційної роботи магістра… **ЗАТВЕРДЖЕНО** кафедрою ПІ протокол № 5 від 13.11.2023р. 3.2 Оформлення пояснювальної записки згідно з ДСТУ 3008:2015 Звіти у сфері науки і техніки. Структура та правила оформлювання. **Шаблон** затверджений засіданням кафедри №3 від 16.10.2023.	Кількість сторінок (рисунків, таблиць, джерел) заявлених в рефераті повинна співпадати з кількістю сторінок (рисунків, таблиць, джерел) в записці.	4

Експерт _____ (підпис) Вадим НЕЧВОЛОД (прізвище, ініціали)

11.06.2024

Рисунок Е.3 - Експертний висновок результатів перевірки кваліфікаційної роботи на відповідність оформлення вимогам ДСТУ 3008:2015