# СУВОРОВ А. И., РОЧЕВ К. В. АСПЕКТНО-ОРИЕНТИРОВАННОЕ ПРОФИЛИРОВАНИЕ В .NET: ПОВЫШЕНИЕ ПРОИЗВОДИТЕЛЬНОСТИ И МОНИТОРИНГ НАГРУЗКИ ПО С ПОМОЩЬЮ KPROFILE

*УДК 004.382.2-027.21, ГРНТИ 50.07.05*

Аспектно-ориентированное профилирование в .NET: повышение производительности и мониторинг нагрузки ПО с помощью KProfile

**А. В. Суворов, К. В. Рочев**

Ухтинский государственный технический университет, г. Ухта

*Статья посвящена разработке и применению библиотеки KProfile для аспектно-ориентированного профилирования в .NET. Основная цель KProfile — повышение производительности и мониторинг нагрузки программного обеспечения с минимальной нагрузкой на анализируемую систему. В статье рассматриваются основные принципы аспектно-ориентированного подхода, преимущества его использования для точечного профилирования, а также структура библиотеки KProfile, обеспечивающая легкую интеграцию с различными типами приложений: мобильными, десктопными и веб-системами. Описаны архитектура решения, основные функции библиотеки и примеры применения, демонстрирующие эффективность профилирования и выявления узких мест в коде для улучшения быстродействия приложений.*

***Ключевые слова:*** *Профилирование, Microsoft Visual Studio, C#, WPF, NuGet, Mongo DB, Web API, ASP.NET, Git*

Aspect-Oriented Profiling in .NET: Enhancing Performance and Monitoring Application Load with KProfile

**A. B. Suvorov, K. V. Rochev**

Ukhta State Techincal University, Ukhta

*The article is dedicated to the development and application of the KProfile library for aspect-oriented profiling in .NET. The primary goal of KProfile is to enhance performance and monitor application load with minimal impact on the analyzed system. The article discusses the fundamental principles of the aspect-oriented approach, the benefits of using it for targeted profiling, and the structure of the KProfile library, which ensures easy integration with various types of applications: mobile, desktop, and web systems. The solution's architecture, key library functions, and usage examples are described, demonstrating the effectiveness of profiling and identifying bottlenecks in code to improve application performance.*

***Keywords:*** *Profiling, Microsoft Visual Studio, C#, WPF, NuGet, Mongo DB, Web API, ASP.NET, Git*

## Введение

На протяжении долгих лет существования компьютерной эпохи активно существовал закон Мура, в соответствии с которым каждые два года удваивалось количество процессоров и до позапрошлого десятилетия это позволяло увеличивать тактовую частоту процессоров и, соответственно, вычисление одноядерных систем [1, 2]. Сегодня рост тактовой частоты стал ограниченным из-за достижения границ 14.7-нанометровых технологий в кремниевых процессорах. Это привело к тому, что дальнейшее увеличение мощности через повышение частоты уже невозможно [3]. Закон Мура отчасти продолжает существовать, но за счёт увеличения количества процессоров [4], а написание многопоточных приложений с параллельными расчетами требует дополнительных ресурсов на разработку и синхронизацию. В то же время объемы данных и сложность необходимых вычислений продолжают неуклонно расти, что требует дополнительного внимания к оптимизации программных систем.

Для увеличения производительности программного обеспечения используются различные методы. Наиболее распространёнными из них являются:

1) оптимизация кода средствами разработки, применяемая на разных фазах компиляции [1];
2) механизмы распараллеливания задач для их выполнения на нескольких ядрах вычислительной системы одновременно [2];
3) профилирование быстродействия программного кода на основе инструментальных средств, предоставляемых средой разработки [3];
4) профилирование с помощью собственной реализации замеров быстродействия под конкретную задачу [4];
5) написание эффективных алгоритмов, правильное использование структур данных и функций над ними, которое, зачастую, требует наибольших знаний и предварительных исследований [5].

## Актуальность

Бурное развитие ИТ-рынка приводит к дефициту высококвалифицированных кадров, а при том, что у ряда программистов еще осталась привычка к постоянному росту производительности оборудования, современные программные продукты часто становятся все более сложными и требовательными к ресурсам. При этом распространяются глобальные сервисы, которым требуется обработка огромных объемов данных, распространяются мобильные устройства с высокими требованиями к энергоэффективности и, зачастую, с низкой производительностью. Кроме того, любые изменения в уже оптимизированном коде могут привести к значительному снижению производительности приложения, особенно под большой нагрузкой. Для диагностики и исправления таких проблем используются профилировщики, которые помогают выявить узкие места в коде и оптимизировать его эффективность.

### Цель и задачи

Цель проекта заключается в разработке и реализации программного решения, которое позволит эффективно отслеживать и анализировать производительность различных приложений. Это включает в себя сбор данных о времени отклика, ошибок и других параметров, которые могут влиять на качество работы приложения.

Основной задачей такого комплекса является обеспечение возможности для разработчиков и системных администраторов получать актуальную информацию о состоянии приложений в реальном времени, что позволит своевременно выявлять и устранять проблемы, влияющие на их работу.

### Аналоги

Есть несколько видов профилировщиков:

- Встроенные в среду разработки – они позволяют во время отладки удобно смотреть и проверять код, и видеть проблемы с производительностью, но не позволяют мониторить выполнение системы, поэтому не удобно каждый раз профилировать перед отправкой изменений на сервер, особенно если много разработчиков.
- Позволяющие профилировать код во время выполнения (хорошим примером такого профилировщика является dotTrace от Jetbrainсe. Он позволяет анализировать производительность ПО во время выполнения. Основным его недостатком является большая нагрузка на профилируемое приложение и существенные объемы собираемых данных – несколько гигабайт в час).

Также существует очень похожая система мониторинга производительности от Firebase. Он является прямым аналогом и позволяет мониторить систему в режиме реального времени и выявлять существующие проблемы с производительностью, но работает только на Android или iOS. Таким образом, полноценных конкурентов на глобальном рынке не найдено. Поэтому было решено создать инструмент, который позволяет мониторить быстродействие приложений в режиме онлайн с минимальной нагрузкой на анализируемое приложение.

### Определение границ и функций системы

Для определения границ системы и основных ее функций была разработана контекстная диаграмма, представляющая как будет выглядеть процесс анализа быстродействия приложений. В данной модели участвуют 3 внешние сущности: разработчик, администратор и измеряемая система, которые взаимодействуют в процессе между собой (Рисунок 1).

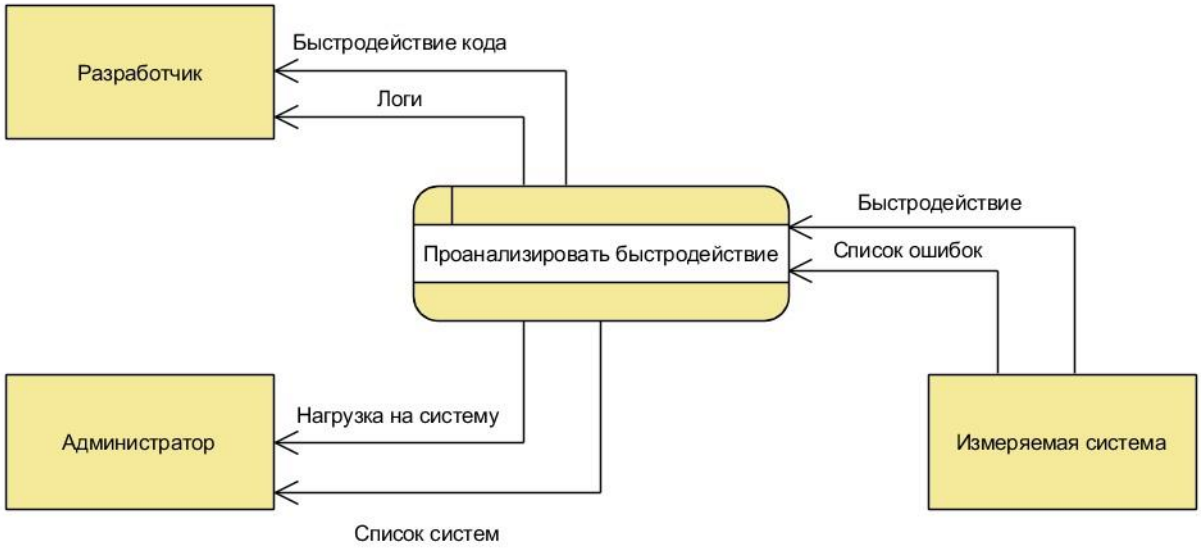


Рисунок 1. Контекстная диаграмма

Основной компонент предлагаемого решения – это библиотека для профилирования (на данный момент реализована на C#). Она подключается к измеряемой системе и передает сервису профилировщика сведения о быстродействии и ошибках приложения. В дальнейшем из собранных наборов данных формируются отчеты, которые может посмотреть администратор, анализирующий общее состояние и нагрузку на приложение. В случае проблем с производительностью или при наличии ошибок к работе подключается разработчик, которому доступны подробные отчеты об ошибках и сведения о быстродействии отдельных функций измеряемой системы (Рисунок 2).

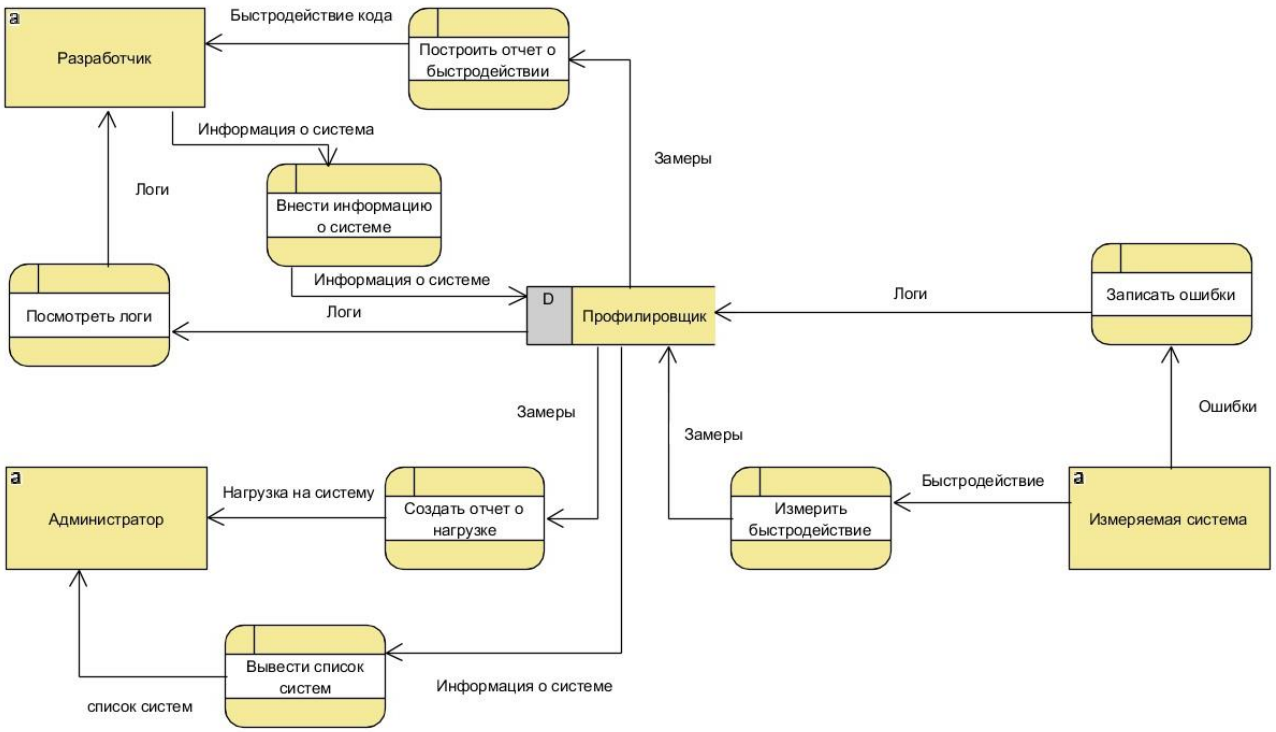


Рисунок 2. Системная диаграмма

Таким образом выделены следующие функциональные требования:

1. Ввод данных о системе
2. Измерение быстродействия
3. Записать ошибки
4. Построение отчета о быстродействии
5. Сформировать отчет об ошибках
6. Создание отчета о нагрузке

**Архитектура проекта**

Основными функциями системы являются сбор данных о быстродействии и ошибках, их вывод и анализ. Для того чтобы обеспечить эти возможности, была разработана библиотека профилирования KProfile, которая является ключевым компонентом системы и подключается к различным ее составляющим. Она представлена как пакет, распространяемый в магазине компонентов NuGet для Visual Studio соответственно с размещением сведений об этом пакете на Git.

Этот компонент подключается через NuGet пакеты к приложению пользователя и позволяет мониторить производительность с использованием аспектно-ориентированного подхода. Также эта библиотека является составной частью веб сервиса и десктопного приложения для анализа данных. В качестве средств разработки использовались Visual Studio, WPF для десктопного приложения, ASP.NET для разработки сервиса сбора и анализа данных и веб клиента, показывающего результаты профилирования по разным приложениям (Рисунок 3).

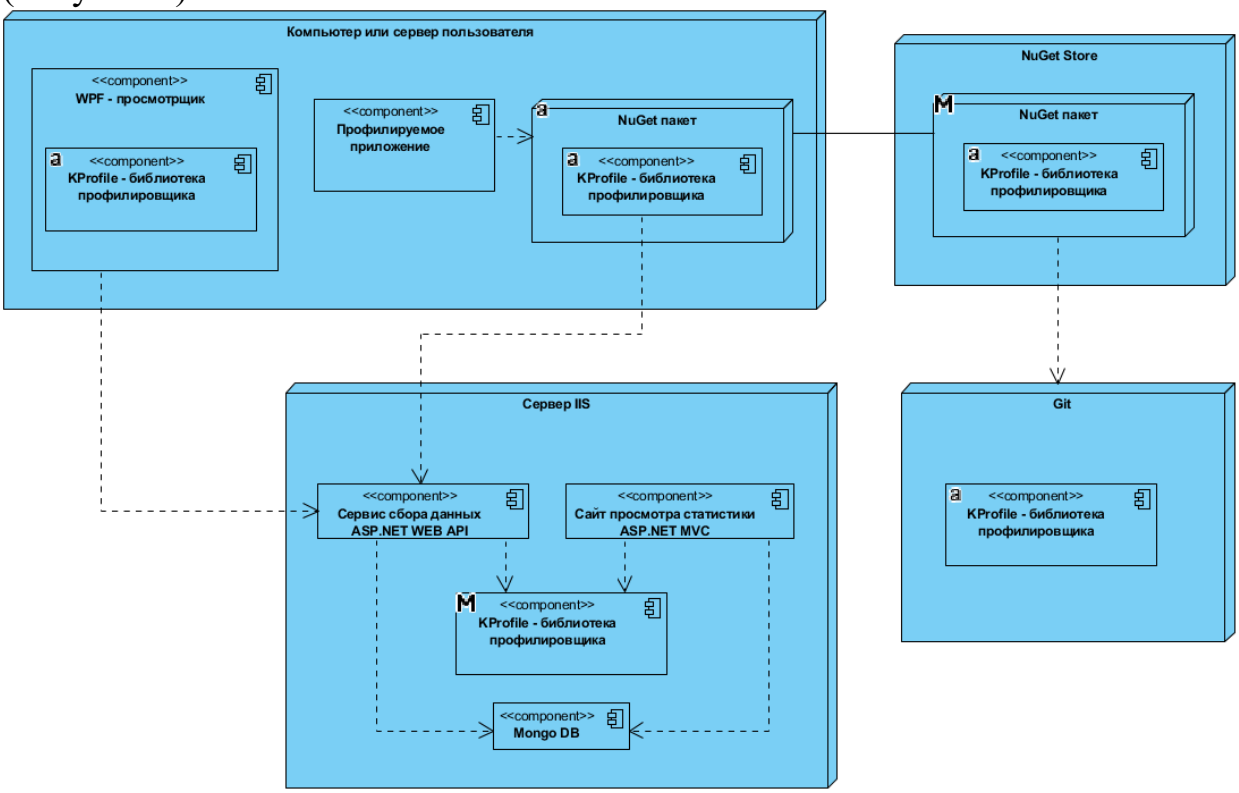


Рисунок 3. Архитектура проекта

Для хранения данных в библиотеке KProfile представлен интерфейс IRepo. А также его базовая реализация на основе файловой системы или обращения к

API веб-сервиса. На стороне сервиса он реализуется с помощью репозитариев MongoDB, а пользователь библиотеки может реализовать его и любым иным способом – хоть на основе реляционных СУБД.

Логическая модель (Рисунок 4) демонстрирует как данные профилирования представлены в реляционном виде.

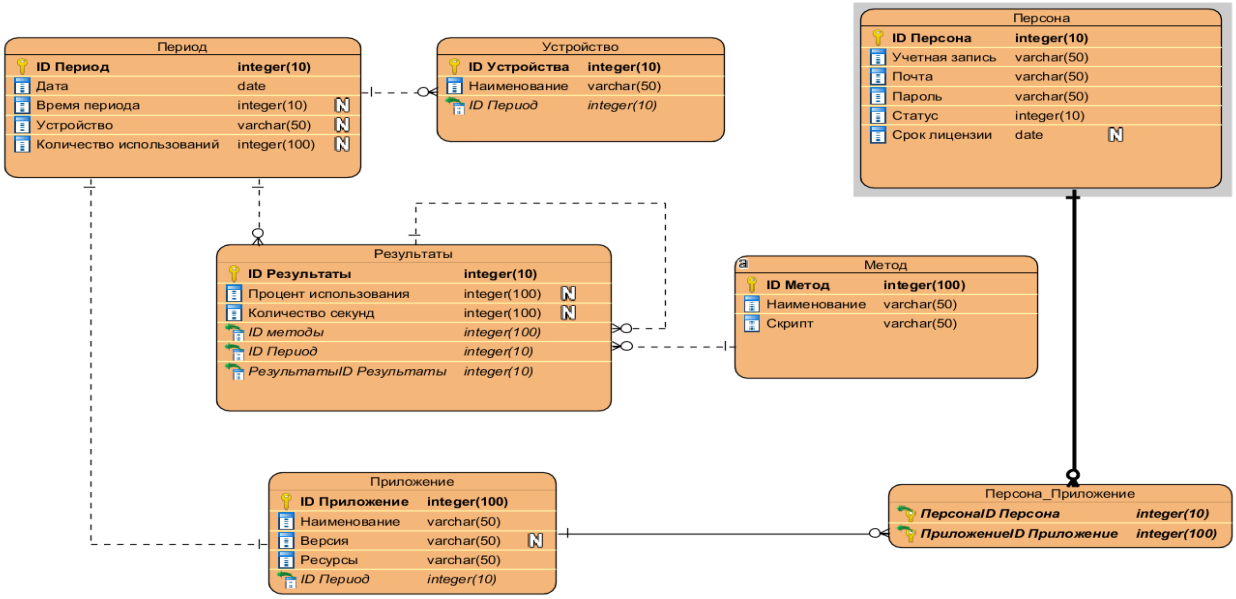


Рисунок 4. Логическая модель

Диаграмма классов (Рисунок 5) демонстрирует структуру основной библиотеки KProfile.

- Хранение данных, которое можно реализовать разными способами: IRepo, Storage.
- Профилирование и запись профилирования: TimeProfiler – основной класс профилировщика, ProfilerAttribute – атрибут для разметки профилируемых классов и функций.
- Логирование ошибок в пользовательском приложении: ErrorLog, ErrorLogger.

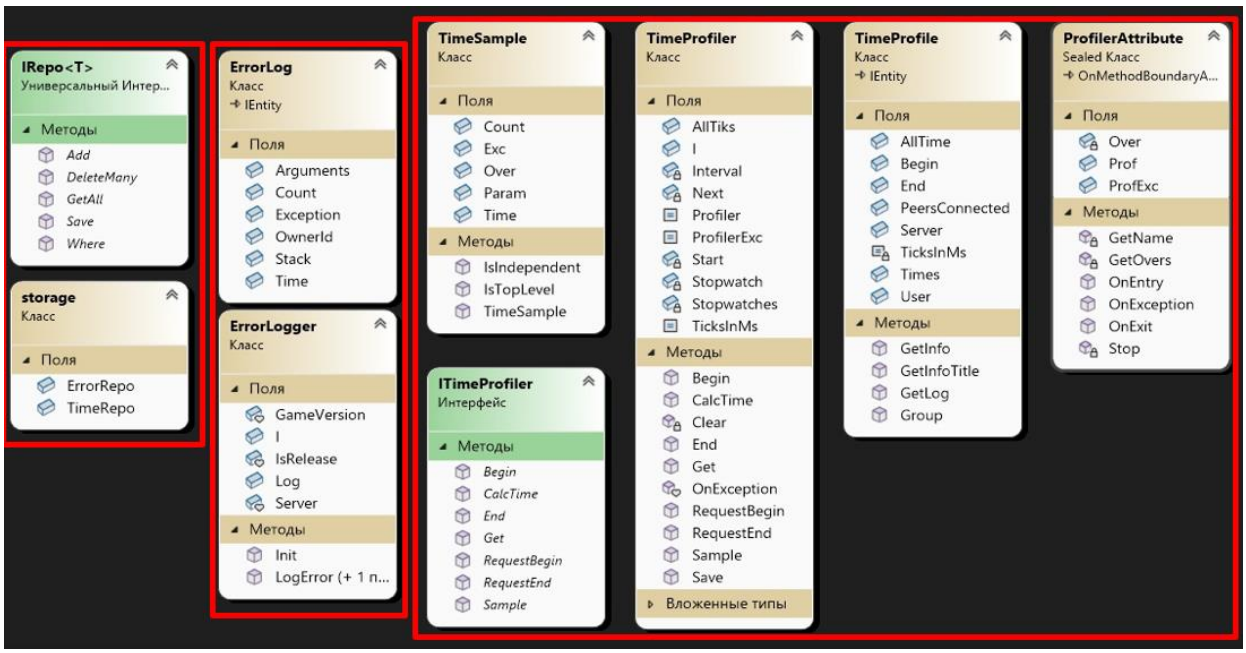


Рисунок 5. Диаграмма класса KProfile

На рисунке 6 показан набор классов WPF приложения для просмотра аналитики профилирования:

- App, mainwindow, uihelper – интерфейсная часть
- Profileranalizer, datapoint, errorinfo – инструменты для анализа профилирования
- Filerepo – пользовательская реализация локального способа хранения данных

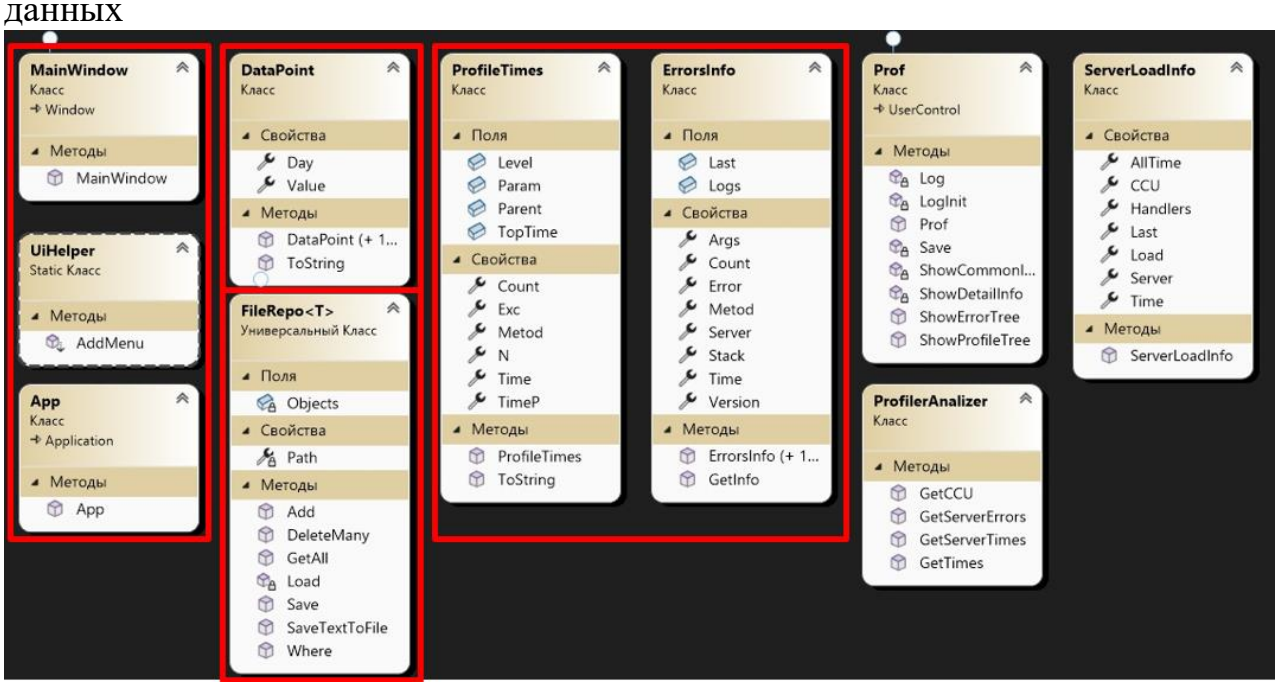


Рисунок 6. Диаграмма класса KProfile.WPF

Код библиотеки доступен для изучения и просмотра клиенту, также он может установить себе локальное приложение (Рисунок 6, 7), чтобы не отправлять результаты профилирования на наш сервис, а использовать

локальное хранение, если в профилируемых приложениях будут какие-либо конфиденциальные данные для измерения.

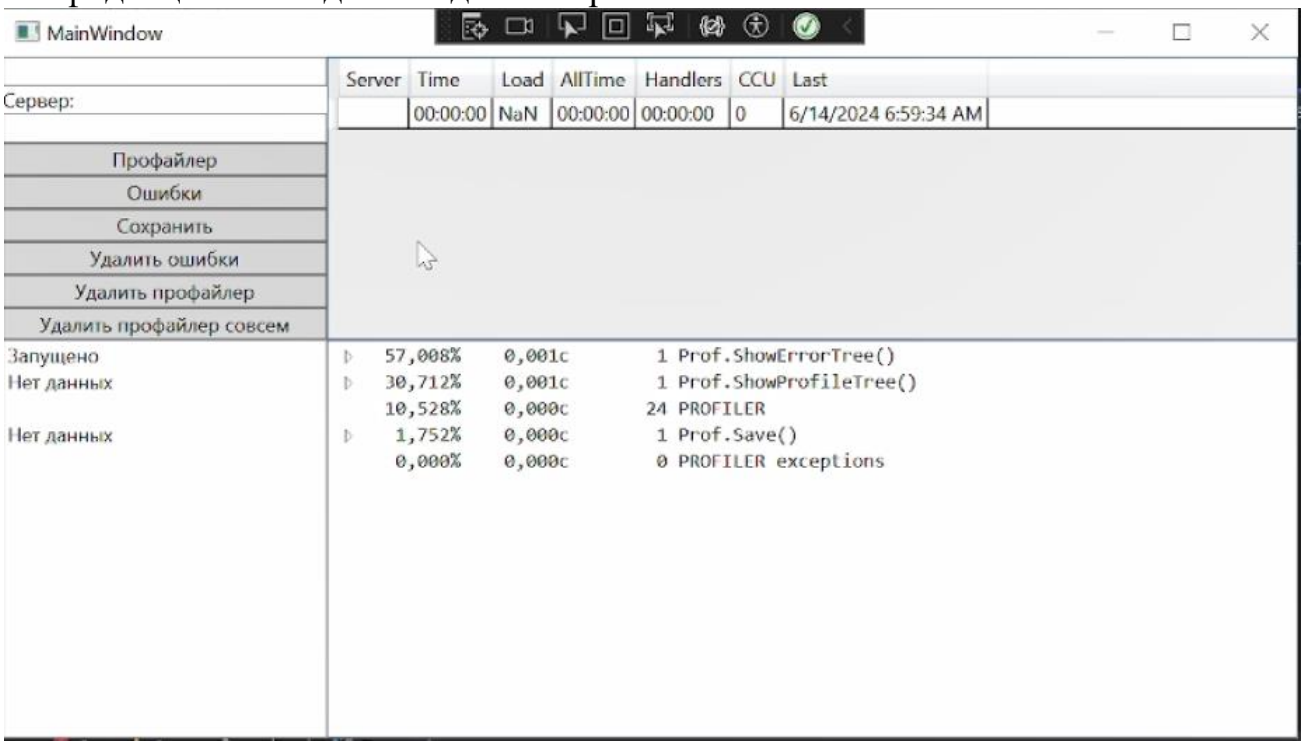


Рисунок 7. Интерфейс прототипа локального просмотрщика результатов профилирования

Для использования профилировщика требуется скачать его из хранилища пакетов NuGet с помощью графического интерфейса лили с помощью команды *Install-Package KProfiler*.

После этого можно будет отмечать атрибутом [Profiler] функции или классы, которые нас интересуют, что позволяет точечно отслеживать производительность, снижая нагрузку на анализируемую систему и обеспечивая, при этом, удобство разметки за счет аспектно-ориенированного подхода.

```
[Profiler] // либо для всего класса – будут замерены все его методы
[ApiController]
public class FController : ControllerBase
{
   [Profiler] // либо для отдельных методов и всего
   public string Get()
   {
      // код, быстродействие которого измеряется
   }
}
```

**Заключение**

В заключении хотелось бы сказать, что в проекте была предпринята попытка реализовать компонент магазина пакетов NuGet для Visual Studio, который позволяет анализировать быстродействие приложений, достаточно

легко интегрируется и мало влияет на финальную производительность измеряемого приложения. Работа сервиса возможна в любых .NET приложениях, будь то мобильные десктопные или веб системы, что позволит пользователям мониторить, повышать стабильность и улучшать быстродействие своих приложений, как в высоконагруженных серверных случаях, так и в низко производительных мобильных кейсах.

**Список использованных источников и литературы**

1. Вернер В., Кузнецов Е., Сауров А. ЗАКОНУ МУРА 50 ЛЕТ: РАЗВИТИЕ МИКРОНАНОЭЛЕКТРОНИКИ // Наноиндустрия. 2015. № 5 (59). С. 56-73.

2. Назилин В. С., Чернова В. С., Степаненко Д. А. Перспективы развития цифрового рынка: Закон Хуанга против Закона Мура // Вестник образования и развития науки Российской академии естественных наук. 2021. № 1. С. 64-68.

3. ЗАКОН МУРА БОЛЬШЕ НЕ РАБОТАЕТ // Век качества. 2015. № 1. С. 54-55.

4. Зубкова В. В. Анализ актуальности закона МУРА // Перспективы развития информационных технологий. 2014. № 21. С. 136-140.

5. Четверина О. А. Повышение производительности кода при однофазной компиляции // Программирование. – 2016. № 1. – С. 51-59.

6. Джойша П. Г., Шрайбер Р. С., Банерджи П., Боэм Х.-Дж., Чакрабарти Д. Р. О методике прозрачного расширения возможностей классических оптимизаций компилятора для многопоточного кода // Транзакции ACM для языков программирования и систем. – 2012. – Т. 34. № 2.

7. Дараган Е. И. Система анализа производительности программного кода // Известия Тульского государственного университета. Технические науки. – 2013. – № 9-2. – С. 89-94.

8. Рочев К. В., Базарова А. М. Разработка атрибутно-ориентированного профилировщика для анализа быстродействия функций программного кода на языке C# // Информационные технологии в управлении и экономике. 2021. №2. Режим доступа: http://itue.ru/Issue/Article/157

9. Рочев К. В. Анализ быстродействия строковых операций языка C# на разных платформах // Программная инженерия. – 2019. – № 6. – С. 274-280.

10. Адам Фримен – "ASP. NET Core MVC 2 с примерами на C# для профессионалов", 2019. – 396 с.

11. Нейгел К., Ивьен Б., Глинн Дж., Уотсон К. – "C# 4.0 и платформа .NET 4 для профессионалов", 2011. – 890 с.

12. Джозеф Албахари и Бен Албахари – " C# 9.0. Карманный справочник" 2021. – 120 с.

13. Шеннон Брэдшоу, Йон Брэзил, Кристина Ходоров – "MongoDB: полное руководство", 2020. – 31 с.

14. Профилировщик кода [Электронный ресурс]. Режим доступа: https://learn.microsoft.com/ru-ru/visualstudio/profiling/what-is-a-profiler?view=vs-2022 (дата обращения: 24.05.2024)

15. C# [Электронный ресурс]. Режим доступа: https://learn.microsoft.com/ru-ru/visualstudio/profiling/profiling-feature-tour?view=vs-2022 (дата обращения 24.03.2024)
16. Создание и публикация пакета NuGet [Электронный ресурс]. Режим доступа: https://learn.microsoft.com/ru-ru/nuget/quickstart/create-and-publish-a-package-using-visual-studio?tabs=netcore-cli (дата обращения: 26.05.2024)
17. Веб-API [Электронный ресурс]. Режим доступа: https://learn.microsoft.com/ru-ru/aspnet/core/tutorials/first-mongo-app?view=aspnetcore-8.0&tabs=visual-studio (дата обращения: 04.06.2024)
18. NuGet [Электронный ресурс]. Режим доступа: https://learn.microsoft.com/ru-ru/nuget/what-is-nuget (дата обращения: 10.05.2024)
19. MongoDB [Электронный ресурс]. Режим доступа: https://metanit.com/nosql/mongodb/ (дата обращения: 22.05.2024)
20. Работа с C# [Электронный ресурс]. Режим доступа: https://metanit.com/sharp/tutorial/ (дата обращения: 24.04.2024)

**List of references**

1. Werner V., Kuznetsov E., Saurov A. MOORE'S LAW IS 50 YEARS OLD: DEVELOPMENT OF MICRONANNOELECTRONICS // Nanoindustry. 2015. No. 5 (59). pp. 56-73.
2. Nazilin V. S., Chernova V. S., Stepanenko D. A. Prospects for the development of the digital market: Huang's Law versus Moore's Law // Bulletin of Education and Science Development of the Russian Academy of Natural Sciences. 2021. No. 1. P. 64-68.
3. MOORE'S LAW NO LONGER WORKS // Century of Quality. 2015. No. 1. P. 54-55.
4. Zubkova V.V. Analysis of the relevance of the MURA law // Prospects for the development of information technologies. 2014. No. 21. P. 136-140.
5. Chetverina O. A. Increasing code performance with single-phase compilation // Programming. – 2016. No. 1. – P. 51-59.
6. Joysha P. G., Schreiber R. S., Banerjee P., Boehm H.-J., Chakrabarti D. R. On a technique for transparently extending the capabilities of classical compiler optimizations for multithreaded code // ACM Transactions on Programming Languages and Systems . – 2012. – T. 34. No. 2.
7. Daragan E.I. System for analyzing the performance of program code // News of the Tula State University. Technical Sciences. – 2013. – No. 9-2. – pp. 89-94.
8. Rochev K. V., Bazarova A. M. Development of an attribute-oriented profiler for analyzing the performance of program code functions in C# // Information technologies in management and economics. 2021. No. 2. Access mode: http://itue.ru/Issue/Article/157
9. Rochev K. V. Analysis of the performance of string operations in the C# language on different platforms // Software engineering. – 2019. – No. 6. – P. 274-280.

10. Adam Freeman – “ASP.NET Core MVC 2 with examples in C# for professionals”, 2019. – 396 p.

11. Nagel K., Ivien B., Glynn J., Watson K. - “C# 4.0 and the .NET 4 platform for professionals,” 2011. – 890 p.

12. Joseph Albahari and Ben Albahari – “C# 9.0. Pocket Guide” 2021. – 120 p.

13. Shannon Bradshaw, Jon Brazil, Christina Khodorov – “MongoDB: The Complete Guide”, 2020. – 31 p.

14. Code profiler [Electronic resource]. https://learn.microsoft.com/ru-ru/visualstudio/profiling/what-is-a-profiler?view=vs-2022 (access date: 05/24/2024)

15. C# [Electronic resource]. https://learn.microsoft.com/ru-ru/visualstudio/profiling/profiling-feature-tour?view=vs-2022 (access date 03.24.2024)

16. Creation and publication of a NuGet package [Electronic resource]. https://learn.microsoft.com/ru-ru/nuget/quickstart/create-and-publish-a-package-using-visual-studio?tabs=netcore-cli (access date: 05/26/2024).

17. Web API [Electronic resource]. https://learn.microsoft.com/ru-ru/aspnet/core/tutorials/first-mongo-app?view=aspnetcore-8.0&tabs=visual-studio (access date: 06/04/2024)

18. NuGet [Electronic resource]. https://learn.microsoft.com/ru-ru/nuget/what-is-nuget (access date: 05/10/2024)

19. MongoDB [Electronic resource]. https://metanit.com/nosql/mongodb/ (access date: 05/22/2024)

20. Working with C# [Electronic resource]. https://metanit.com/sharp/tutorial/ (access date: 04/24/2024)