**САМЫЙ БЫСТРЫЙ ШАБЛОНИЗАТОР PHP**

**Я.О. Цвира1**

1 [*tsvira.yarosalv@mail.ru*](mailto:tsvira.yarosalv@mail.ru)

**Аннотация.**

**Введение:** каждый раз на просторах Интернета, в научных сообществах или в обыкновенных беседах, посвященных программированию на PHP, происходят споры по поводу: «Какой шаблонизатор является самым быстрым?» Ответ на этот вопрос, обычно, всегда разный. Поэтому это исследование направлено на разоблачение мифом и отыскание истины.

**Материалы и методы:** для исследования были использованы следующие приложения: Excel (построение графиков и запись полученных замеров), Open Server Panel (локальный сервер), PhpStorm (программа для написания кода на PHP), Composer (пакетный менеджер, который предоставляет средства для управления зависимостями).

**Результаты:** в ходе исследования мы получили двух победителей: BladeOne (в районе 10–100 запросов: победитель на слабой и средней нагрузке) и Smarty (в районе 1000–10000 запросов: победитель на высокой и очень высокой нагрузке), а также шаблонизаторы были разделены на три категории, каждая из которых в свое время делится на три вида:

1. Относительно нагрузки запросов и их количества
   1. Шаблонизаторы, которые слабо зависят от количества запросов (Smarty)
   2. Шаблонизаторы, которые средне зависят от количества запросов (Mustache)
   3. Шаблонизаторы, который сильно зависят от количества запросов (Plates, Latte, Blade, Twig, BladeOne)
2. Внутреннее устройство технологии
   1. Шаблонизаторы, которые используют основы ООП и функционального программирования (Smarty)
   2. Шаблонизаторы, которые выступают основой для написание своего шаблонизатора (Mustache)
   3. Шаблонизаторы, который используют ООП, основы функционального программирования и пространственные имена (Plates, Latte, Blade, Twig, BladeOne)
3. Относительно инициализации и нагрузки запросов:
   1. Шаблонизаторы, которые имеют высокую нагрузку инициализации, но низкую нагрузку запросов (Smarty)
   2. Шаблонизаторы, которые имеют среднюю нагрузку инициализации и среднюю нагрузку запросов (Mustache)
   3. Шаблонизаторы, которые имеют низкую нагрузку инициализации, но высокую нагрузку запросов (Plates, Latte, Blade, Twig, BladeOne)

**Выводы:** время выполнения кода – одно из самых важных факторов. Но оно не должно быть главным: стоит учитывать и современность, и безопасность, и удобство, и потребление той или иной технологии. Как сказал однажды Великий мудрец: «Самая лучшая технология — это та технология, который ты умеешь управлять».

**Ключевые слова:**PHP, Twig, Smarty, Blade, BladeOne, Latte, Plates, Mustache, шаблонизаторы, самый быстрый шаблонизатор PHP

*Автор, ответственный за переписку: Я.О. Цвира,* [*tsvira.yarosalv@mail.ru*](mailto:tsvira.yarosalv@mail.ru)

**ВВЕДЕНИЕ**

Несмотря на то, что PHP уже по своей сути является шаблонизатором, также существует множество других. Главная причина появления новых шаблонизаторов заключается в необходимости разделять логику от представления. Благодаря этому frontend-разработчики могут изменять внешний вид сайта (шаблон), а backend – логику и архитектуру. Выбор шаблонизатора при сайтостроительстве, а точнее при серверном программирование веб-сайтов на РHP, является неотъемлемым и важным фактором, так как чем сложнее становится веб-приложение, тем сложнее им управлять.

Поскольку на протяжении нескольких лет в сообществе веб-программистов наблюдается спор по поводу: «Какой шаблонизатор PHP является самым быстрым», то данное исследование является актуальным. Безусловно, у автора имеется и своя причина, по которой он изучает данную проблему: оптимизации своего личного проекта.

***Полнота литературного обзора – литература***

Исследование является полностью автономным в том плане, что кроме документации PHP и шаблонизаторов дополнительная литература не использовалась:

* Руководство по PHP: <https://www.php.net/manual/ru/index.php>
* Руководство Twig: <https://twig.symfony.com/doc/>
* Руководство Smarty: <https://www.smarty.net/documentation>
* Руководство Blade: <https://laravel.com/docs/8.x/blade>
* Руководство BladeOne: <https://github.com/EFTEC/BladeOne/wiki/BladeOne-Manual>
* Руководство Latte: <https://latte.nette.org/en/guide>
* Руководство Plates: <https://platesphp.com/>
* Руководство Mustache: <https://github.com/bobthecow/mustache.php/wiki>

**МАТЕРИАЛЫ И МЕТОДЫ**

В решении поставленного вопроса главная роль отводится характеристикам компьютера, выступающего хостингом. Так как компьютер автора будет использоваться, как хостинг, то следует узнать процессор компьютера, так как это является главной характеристикой (см. рис. 1).

Изображение выглядит как текст

Автоматически созданное описание

Рис. 1 Характеристики компьютера

Не менее важными являются характеристикой локального хостинга, а точнее версии PHP на которой будет замеряться скорость шаблонизаторов (см. рис. 2).

Изображение выглядит как текст

Автоматически созданное описание

Рис. 2 Характеристики локального хостинга (Open Server: настройки)

Так как для шаблонизаторов нет условной единицы замера, то допустим, что данный код PHP будет являться условной единицей для эксперимента:

<?php

        (object) [

            "code" => 101,

            "message" => "Равным образом рамки и место обучения кадров влечет за собой процесс внедрения и модернизации системы обучения кадров, соответствует насущным потребностям."

        ],

?>

Назовём данную единицу «запросом» для упрощения дальнейшего повествования.

Так как собирать данные вручную – это долгий процесс, воспользуемся доступными нам средствами и создадим файл, со следующей структурой:

<?php

        // Указываем то, что нужно записать в файл

$text = ''.$duration.' ';

// Открываем файл в нужном нам режиме. Нам же, нужно его создать и что-то записать.

$fp = **fopen**("/file.txt", "a");//поэтому используем режим 'w'

// записываем данные в открытый файл

**fwrite**($fp, $text);

//не забываем закрыть файл, это ВАЖНО

**fclose**($fp);

?>

Данный файл назовём *file\_router.php* и будем вызывать его с помощью выражения *include* в *PHP.* С помощью этого файла будем записывать производительность в текстовый файл *file.txt.*

Теперь осталось получить и данные, зафиксировать в Excel и сделать вывод. Для этого воспользуемся функцией «Получить данные из текста».

После базовой настройки, определений понятий и фиксации характеристик мы можем приступать к замеру производительности. При проектировании шаблонов и backend воспользуемся документацией шаблонизаторов.

**РЕЗУЛЬТАТЫ ИССЛЕДОВАНИЯ**

В ходе работы были рассмотрен ряд шаблонизаторов: Twig, Smarty, Blade, BladeOne, Latte, Plates, Mustache. А также PHP для определения «идеального значения».

В ходе исследования можно шаблонизаторы разделить на три вида относительно нагрузки запросов и их количества:

1. Шаблонизаторы, которые слабо зависят от количества запросов (Smarty)
2. Шаблонизаторы, которые средне зависят от количества запросов (Mustache)
3. Шаблонизаторы, который сильно зависят от количества запросов (Plates, Latte, Blade, Twig, BladeOne)

Автору стала интересна полученная зависимость, поэтому хоть анализ шаблонизаторов, как технологии, не является основной задачей, но она стала вытекающей. По этой причине, верхнее распределение можно осуществить по другой системе (внутреннее устройство технологии):

1. Шаблонизаторы, которые используют основы ООП и функционального программирования (Smarty)
2. Шаблонизаторы, которые выступают основой для написание своего шаблонизатора (Mustache)
3. Шаблонизаторы, который используют ООП, основы функционального программирования и пространственные имена (Plates, Latte, Blade, Twig, BladeOne)

Так как главной задачей было выявить самый быстрый шаблонизатор PHP, то составим характеристические таблицы:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Место | 10 запросов | 100 запросов | 1000 запросов | 10000 запросов |
| 1 | BladeOne | BladeOne | Smarty | Smarty |
| 2 | Plates | Plates | Mustache | Mustache |
| 3 | Twig | Mustache | Plates | Plates |
| 4 | Mustache | Smarty | BladeOne | Latte |
| 5 | Blade | Twig | Latte | Blade |
| 6 | Smarty | Blade | Twig | Twig |
| 7 | Latte | Latte | Blade | Blade |

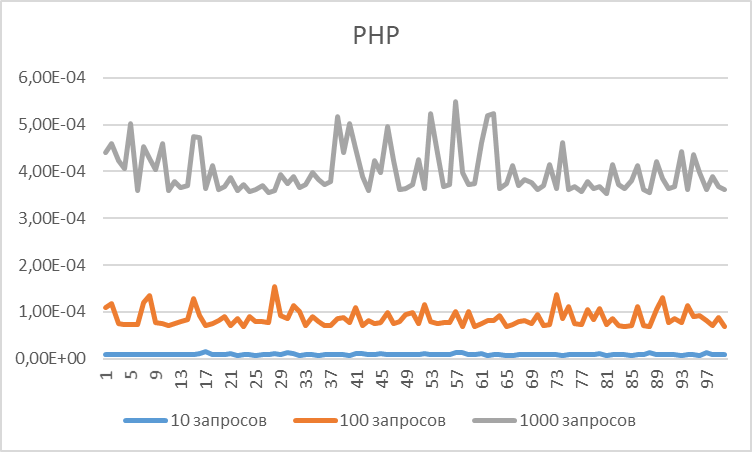
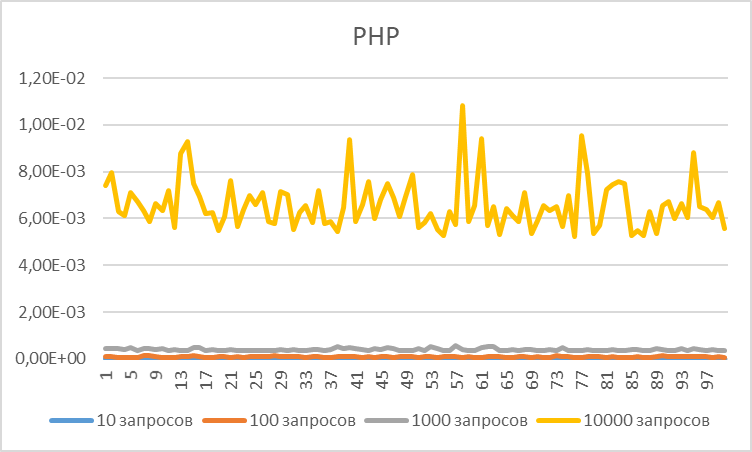
Нельзя не обратить внимание, что у каждого шаблонизатора с количеством запросов растет средняя загрузка веб-страницы, а средняя нагрузка запроса падает. Почему же так? Автор специально в исследовании не взял во внимание инициализацию шаблонизаторов, хоть и инициализация неотъемлемо влияет на скорость, но без предварительных экспериментов ее сложно определить. Теоретически у PHP средней нагрузки запроса должна всегда быть равна, но это при «идеальной системе». Система же наших замеров не является «идеальной» и поэтому возможны допущения, а также влияние других факторов, которые не могут войти в исследование. Поэтому можно сделать такое деление шаблонизаторов на виды (деление относительно инициализации и нагрузки запросов):

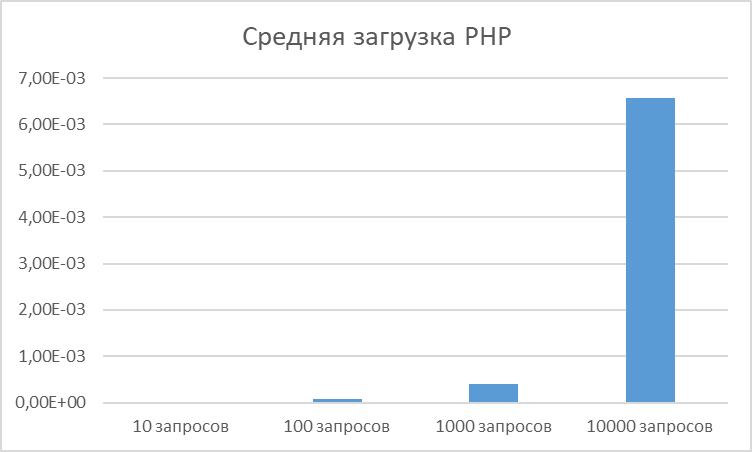
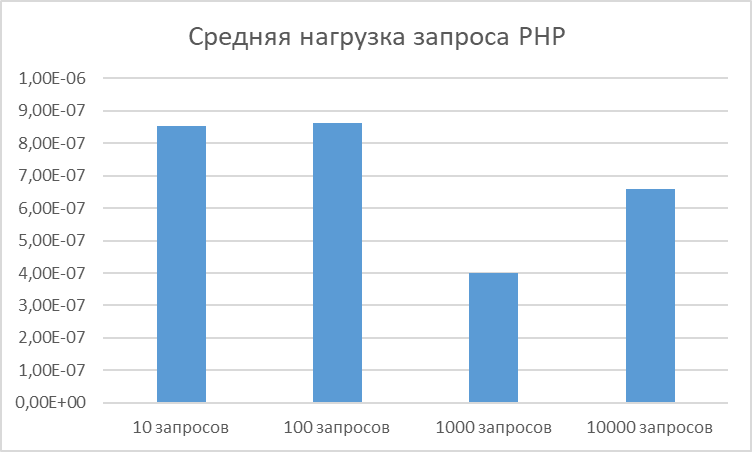
1. Шаблонизаторы, которые имеют высокую нагрузку инициализации, но низкую нагрузку запросов (Smarty)
2. Шаблонизаторы, которые имеют среднюю нагрузку инициализации и среднюю нагрузку запросов (Mustache)
3. Шаблонизаторы, которые имеют низкую нагрузку инициализации, но высокую нагрузку запросов (Plates, Latte, Blade, Twig, BladeOne)

А теперь пришло время ответить на вопрос: «Какой шаблонизатор PHP является самым быстрым?» В ходе исследования мы получили двух победителей: BladeOne (в районе 10–100 запросов: победитель на слабой и средней нагрузке) и Smarty (в районе 1000–10000 запросов: победитель на высокой и очень высокой нагрузке).

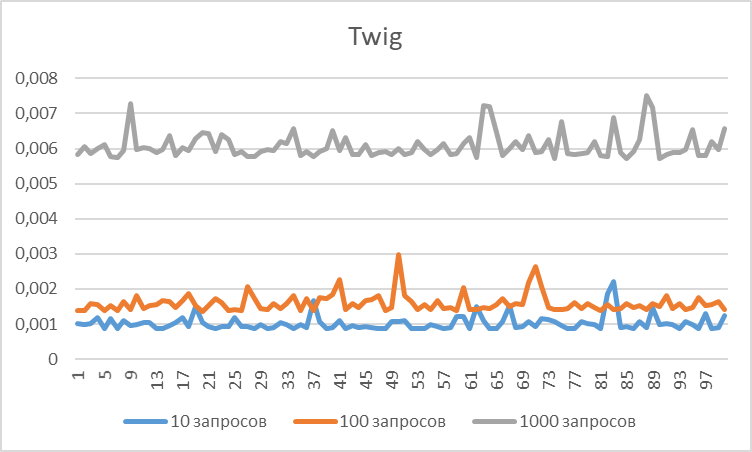
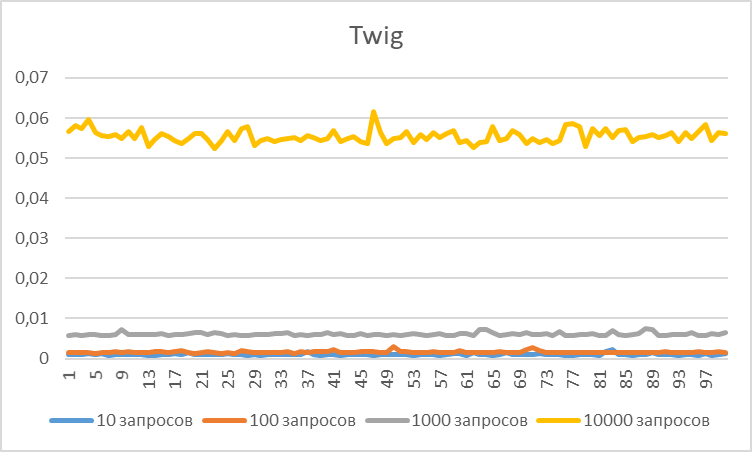
Время выполнения кода – одно из самых важных факторов. Но оно не должно быть главным: стоит учитывать и современность, и безопасность, и удобство, и потребление той или иной технологии. Как сказал однажды Великий мудрец: «Самая лучшая технология — это та технология, который ты умеешь управлять».

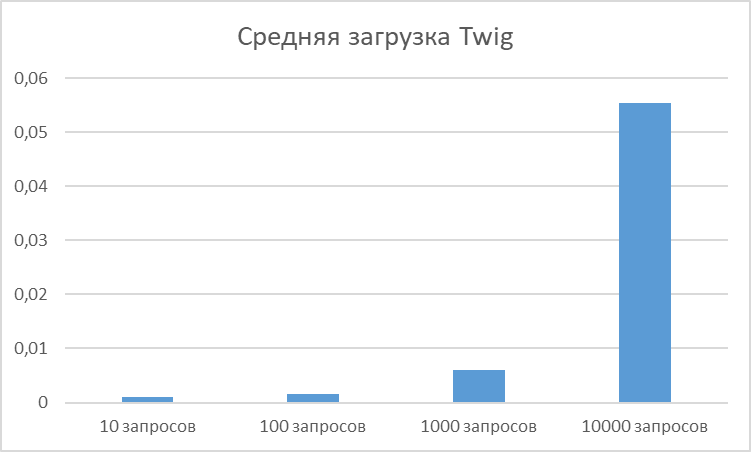
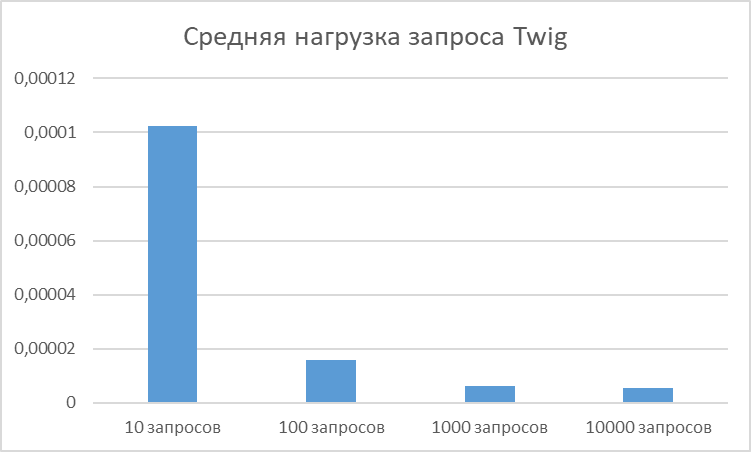
**PHP**

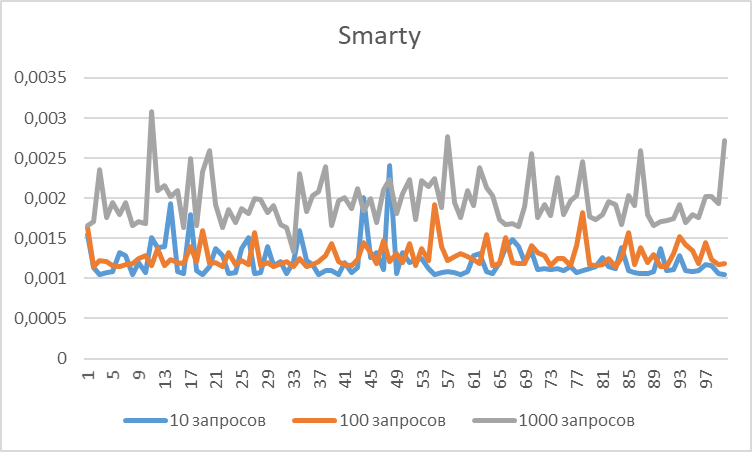
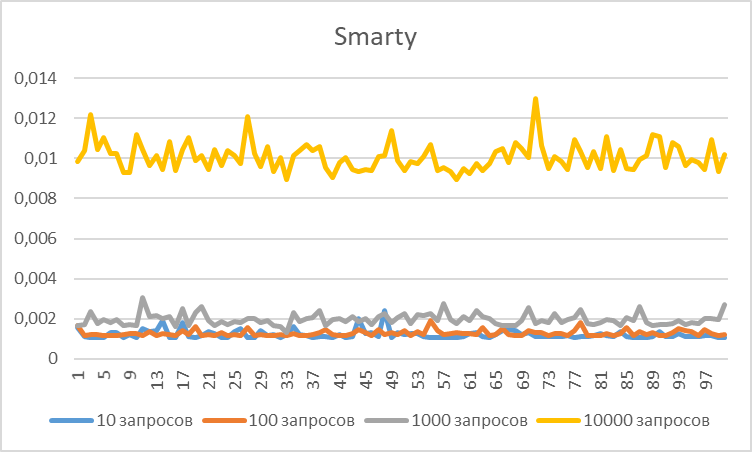
 

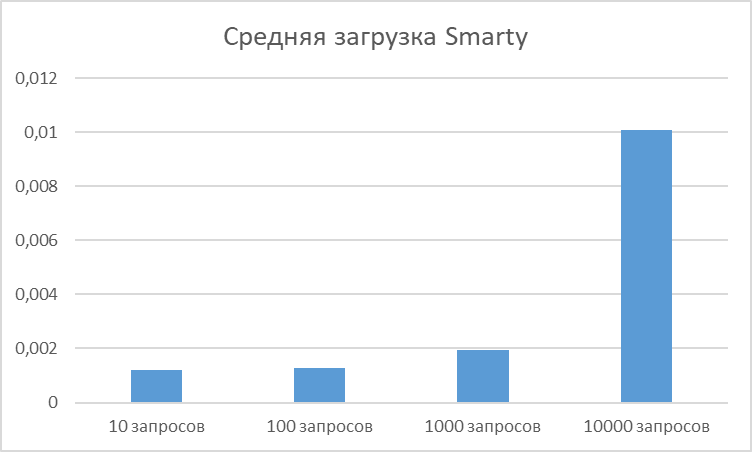
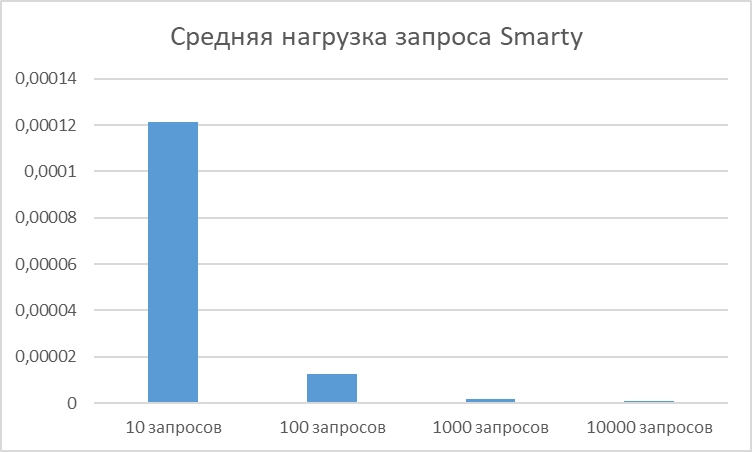
**Twig**

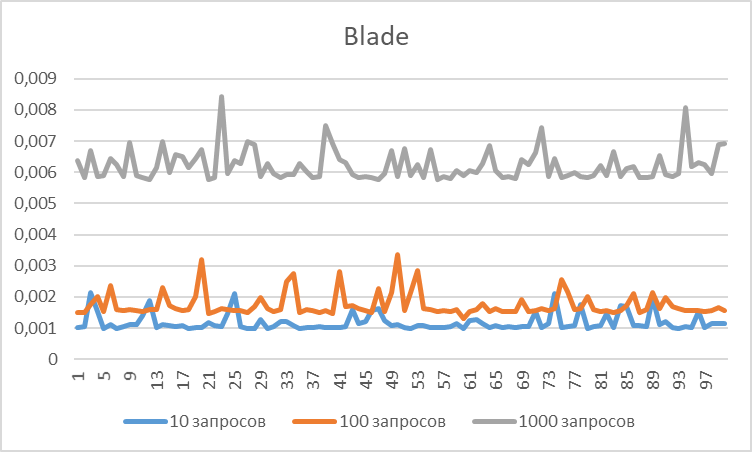
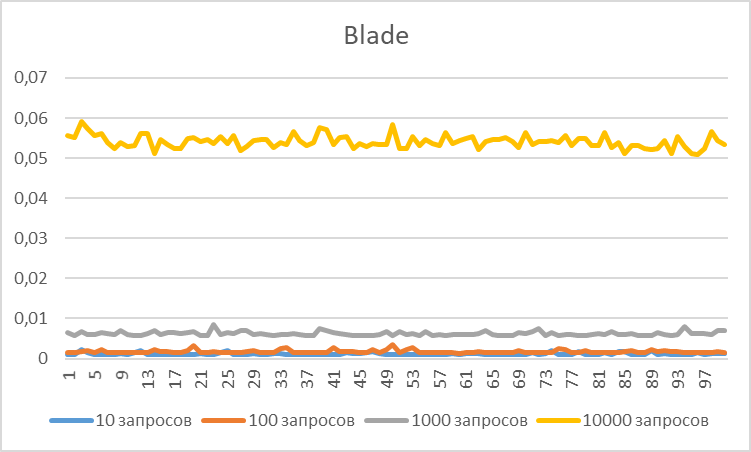
 

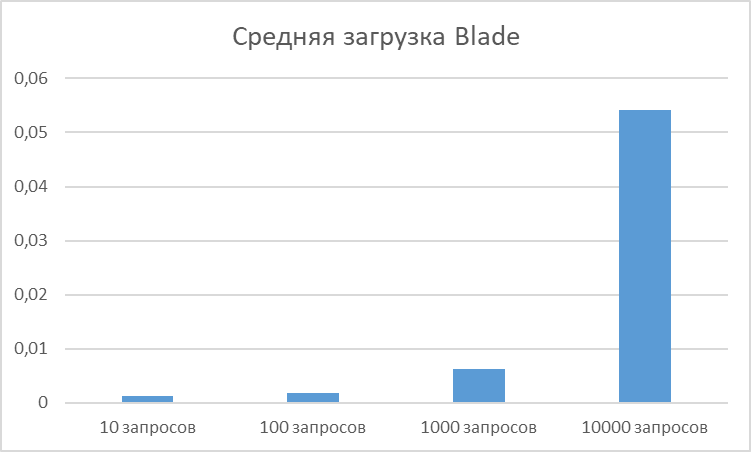
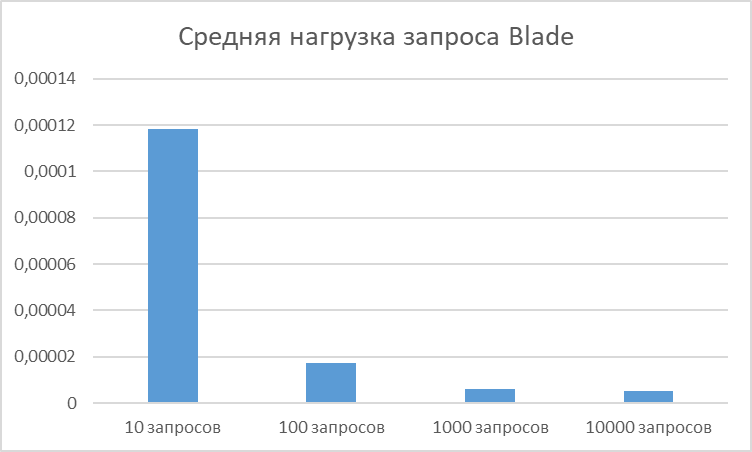
**Smarty**

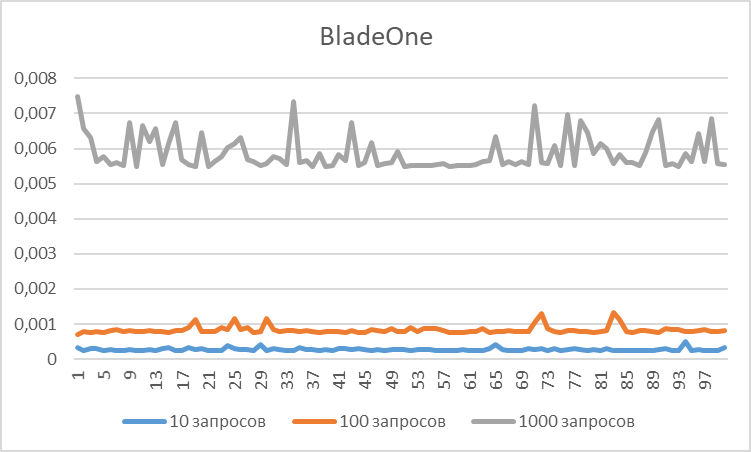
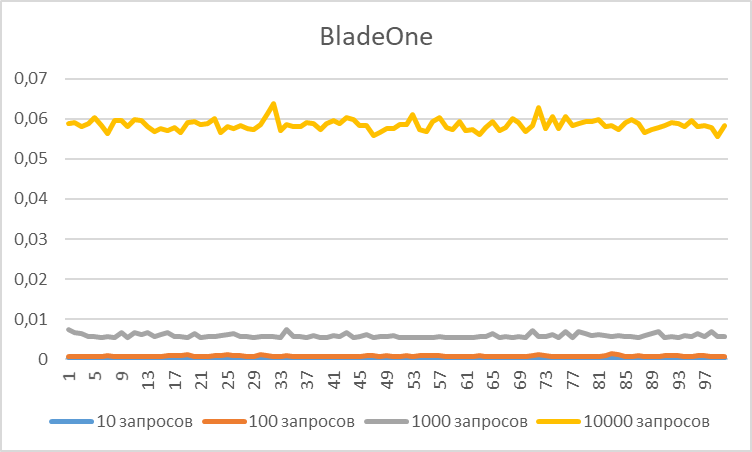
 

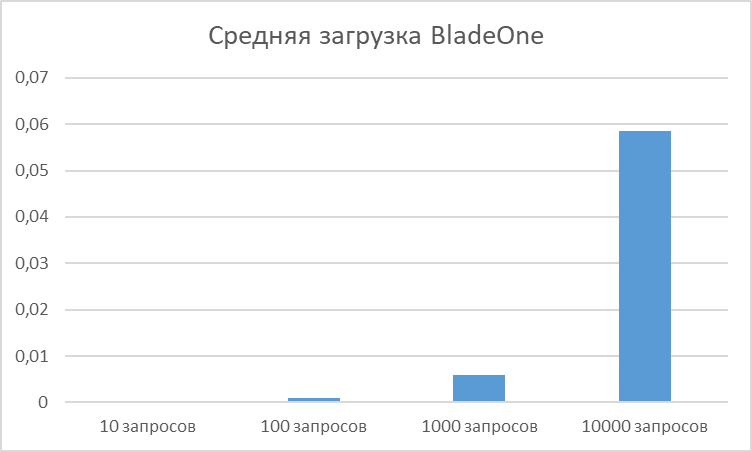
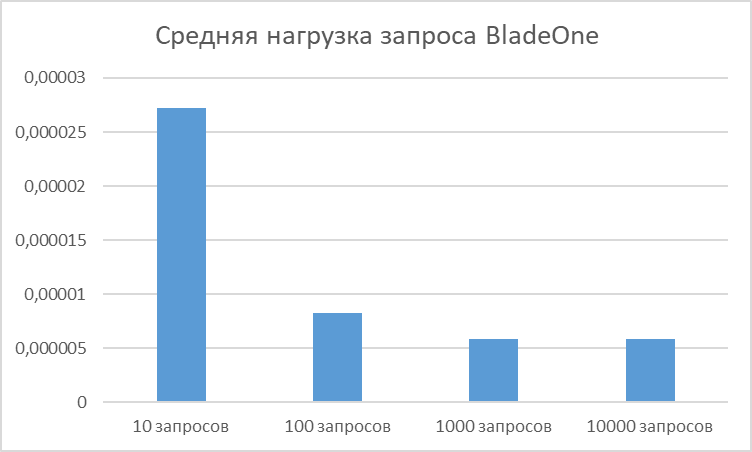
**Blade**

** **

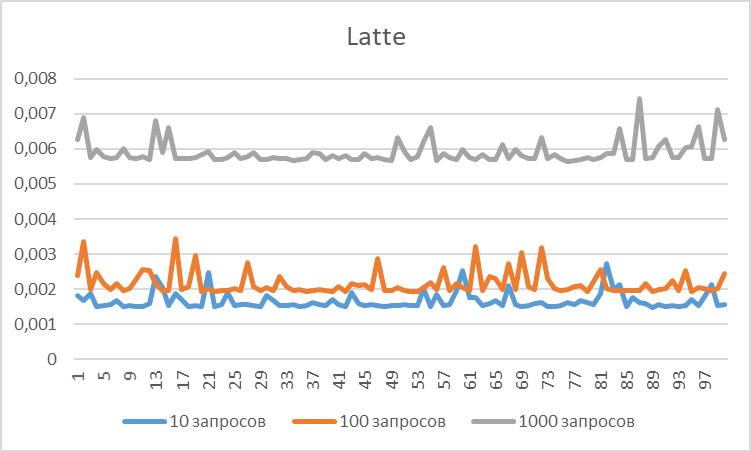
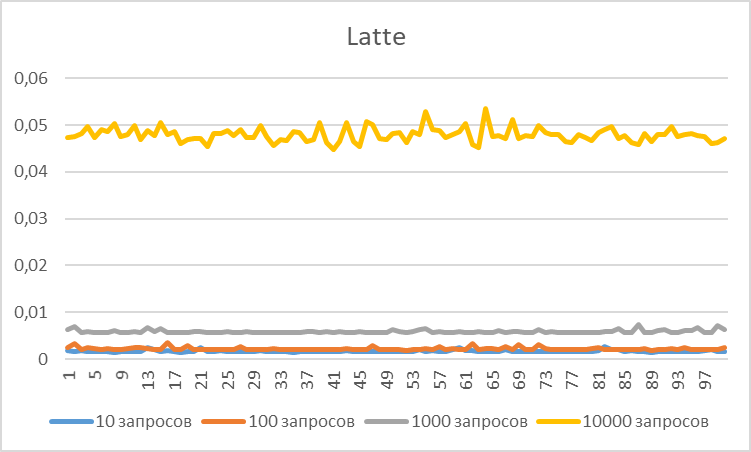
 

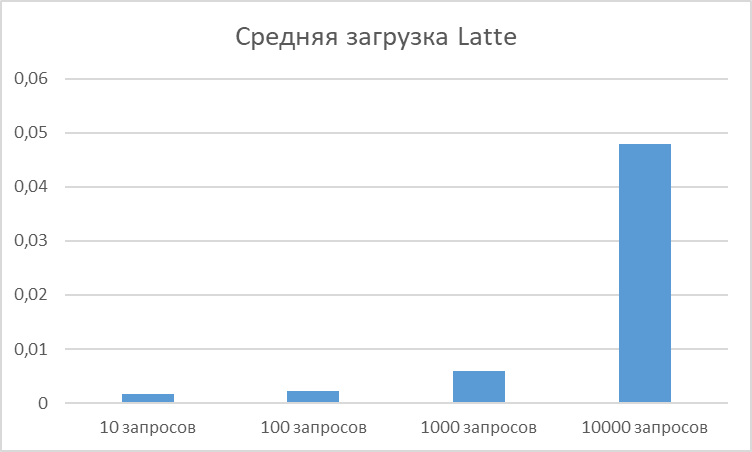
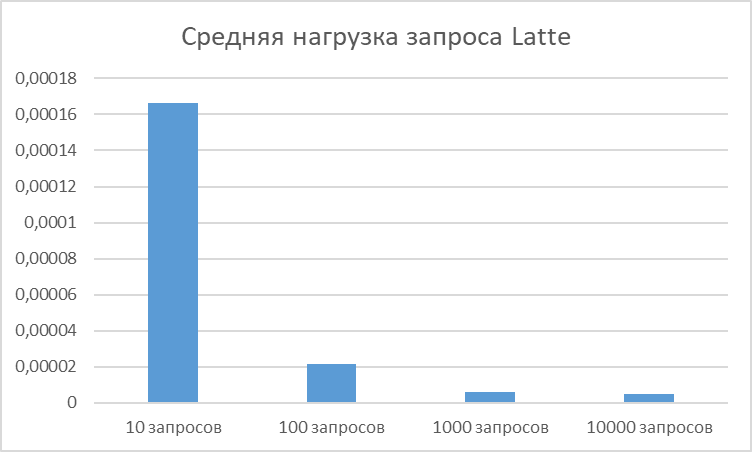
**BladeOne**

** **

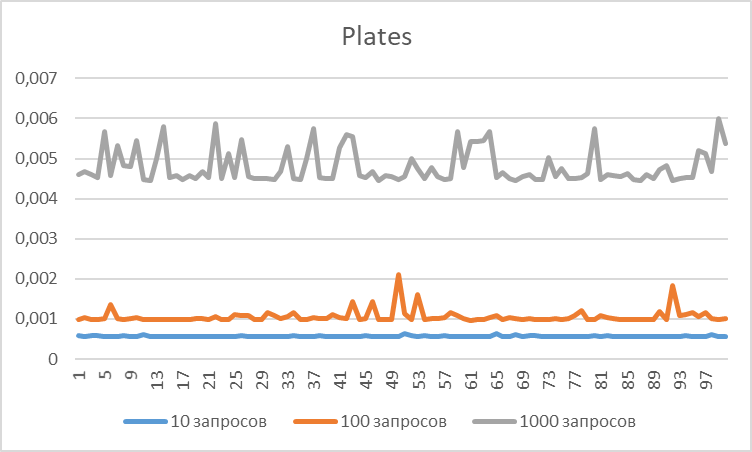
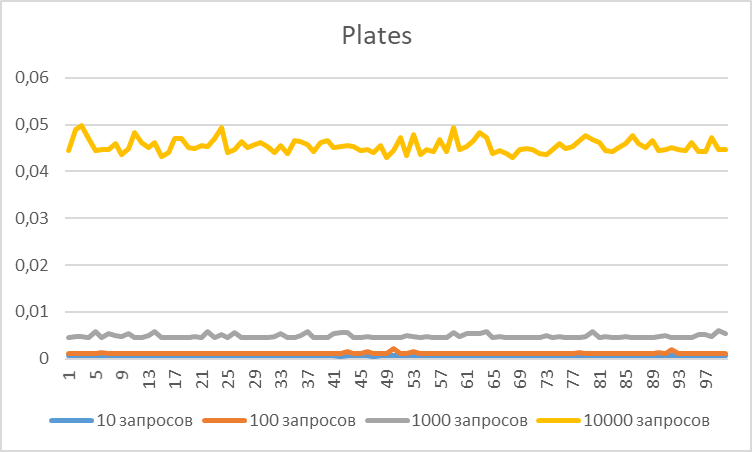
** **

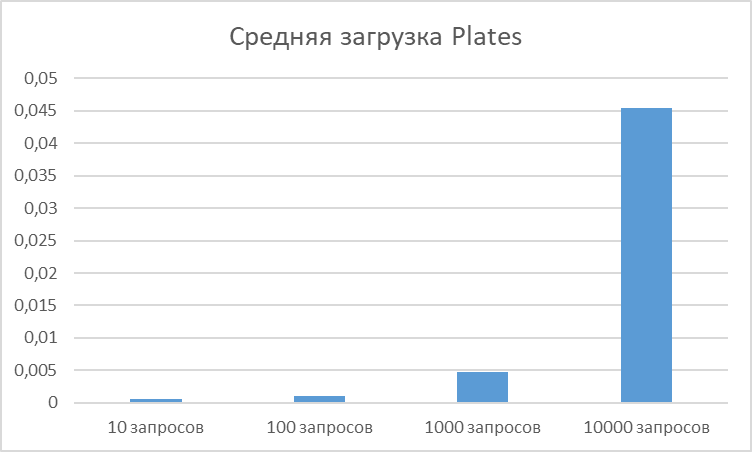
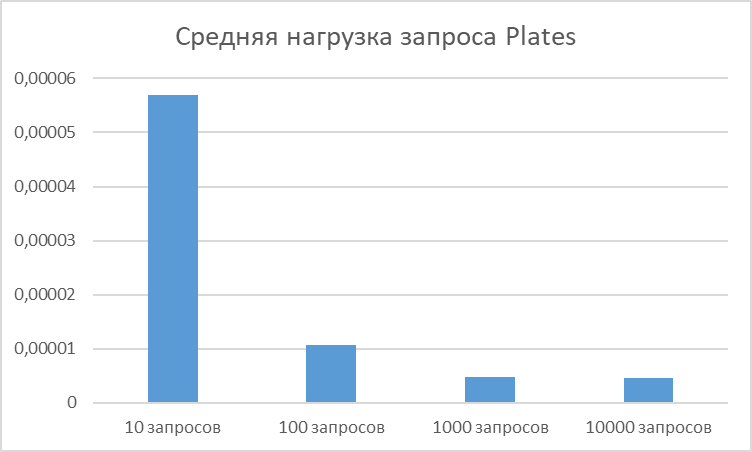
**Latte**

** **

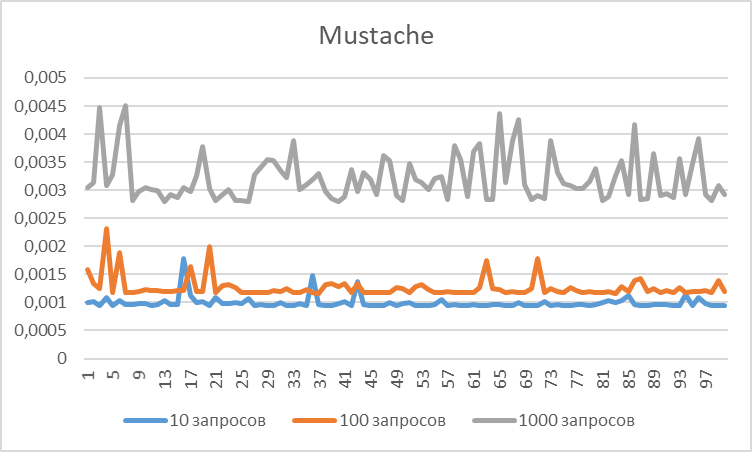
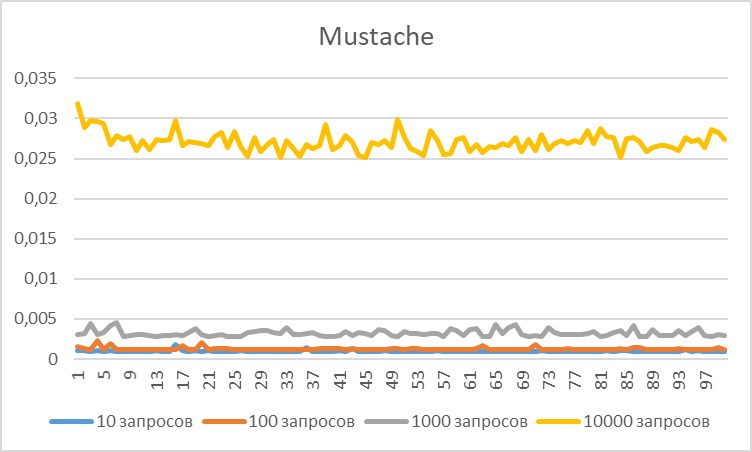
** **

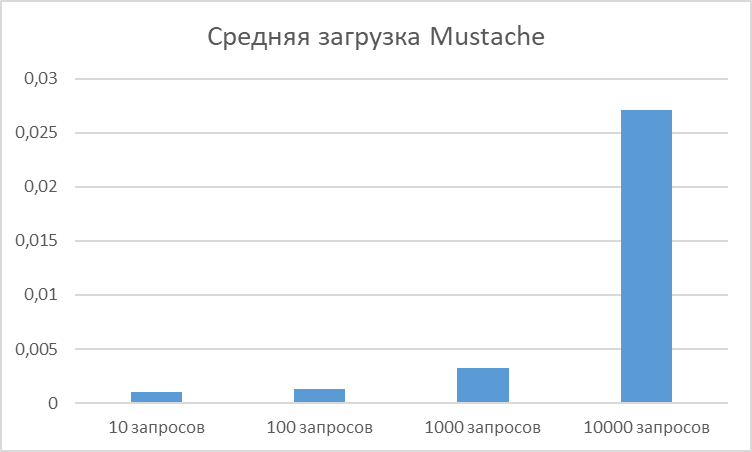
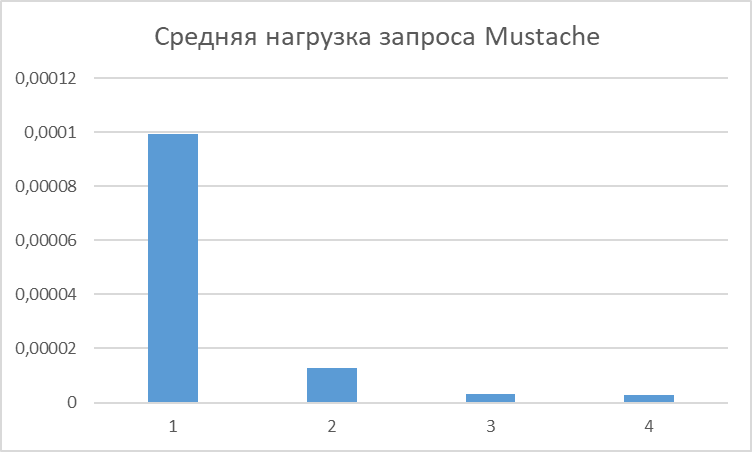
**Plates**

** **

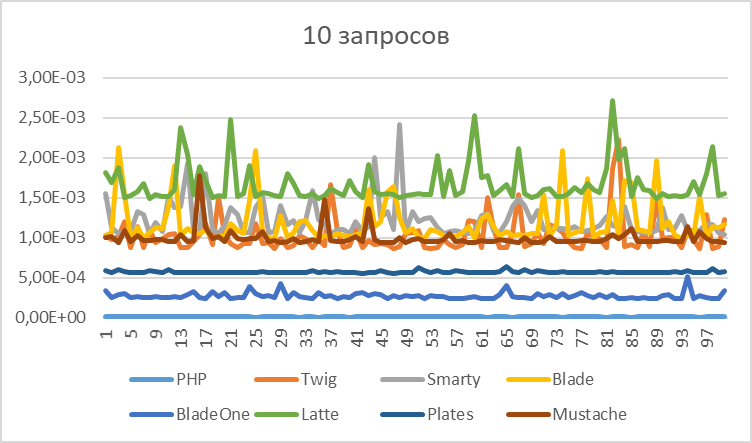
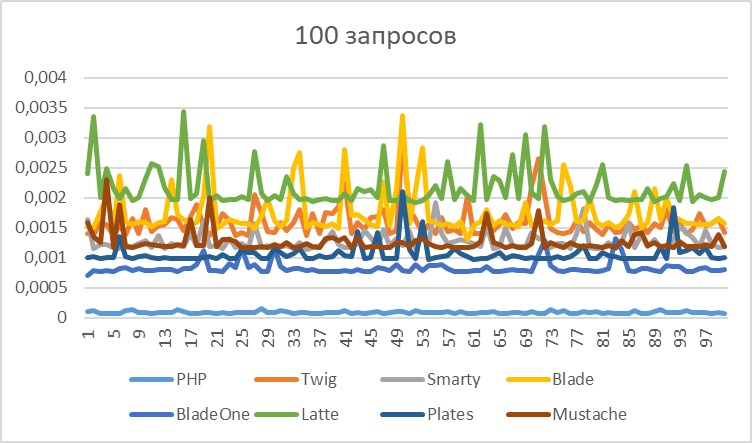
** **

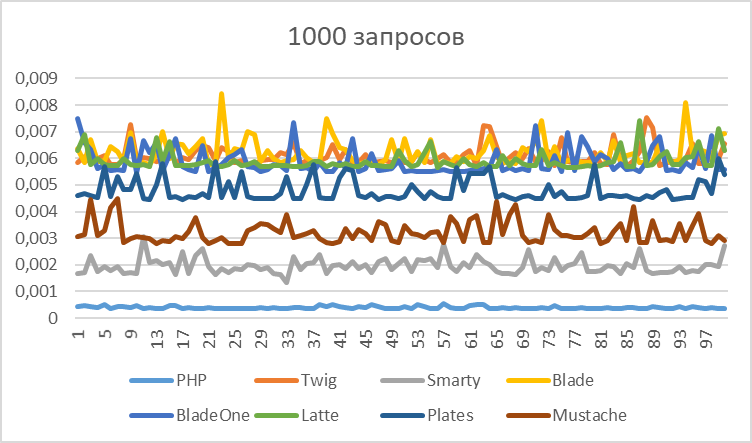
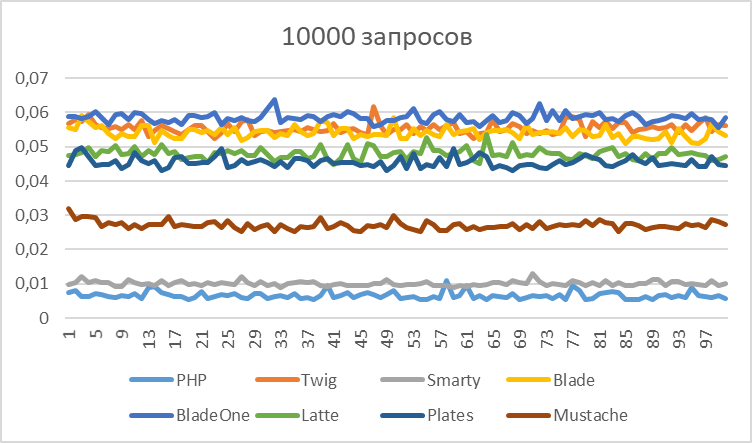
**Mustache**

** **

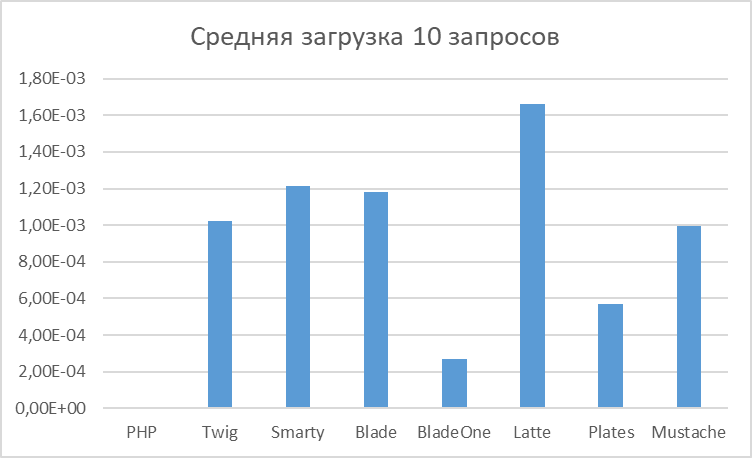
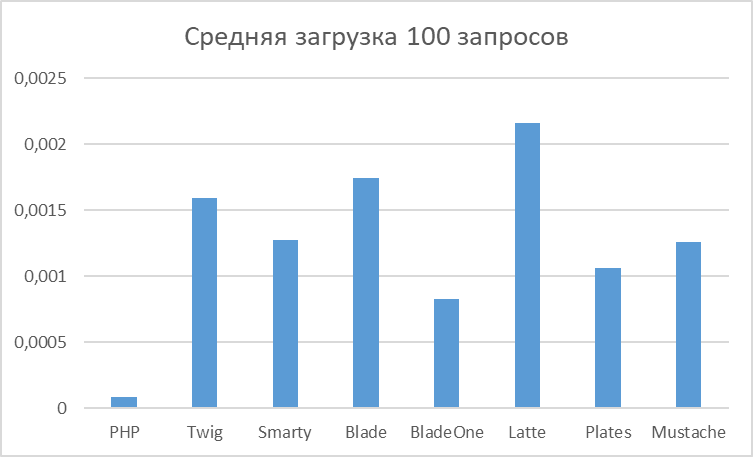
** **

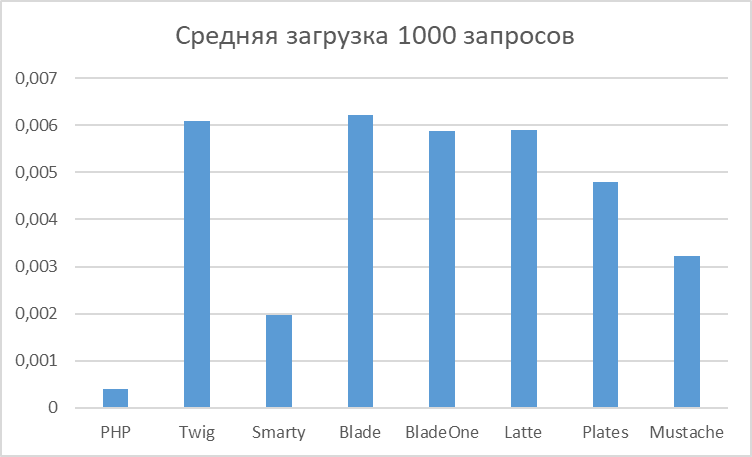
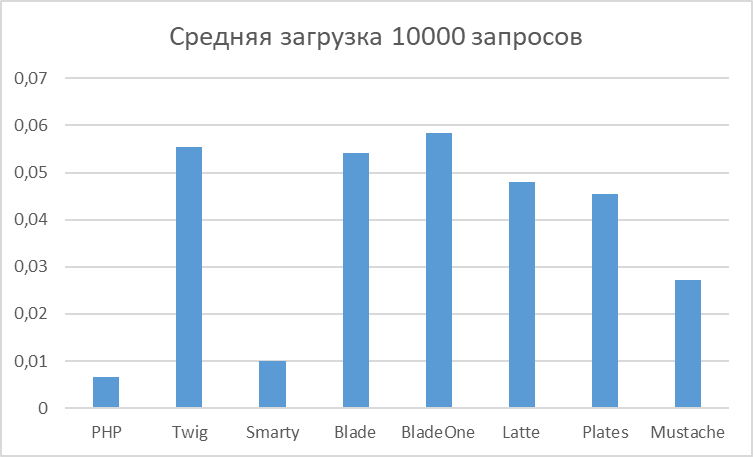
**Обобщение результатов**

** **

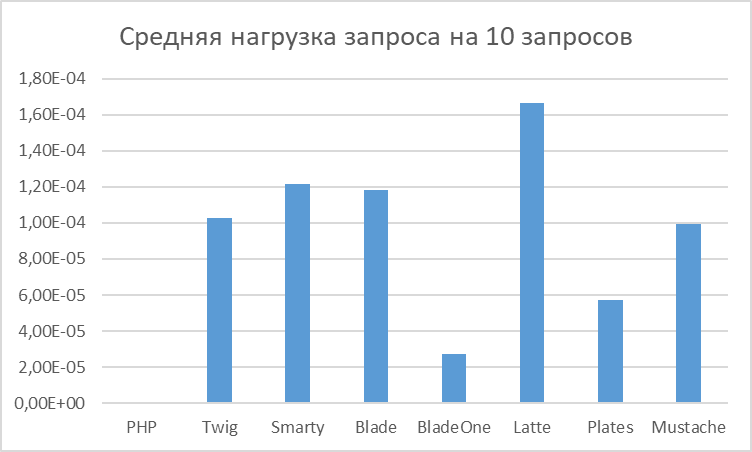
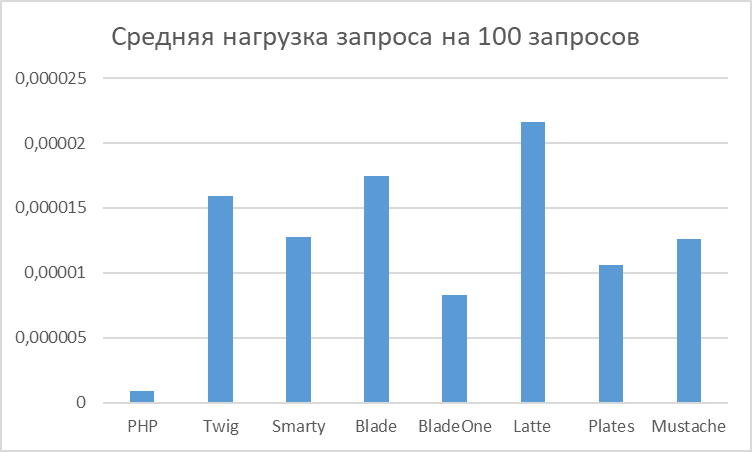
** **

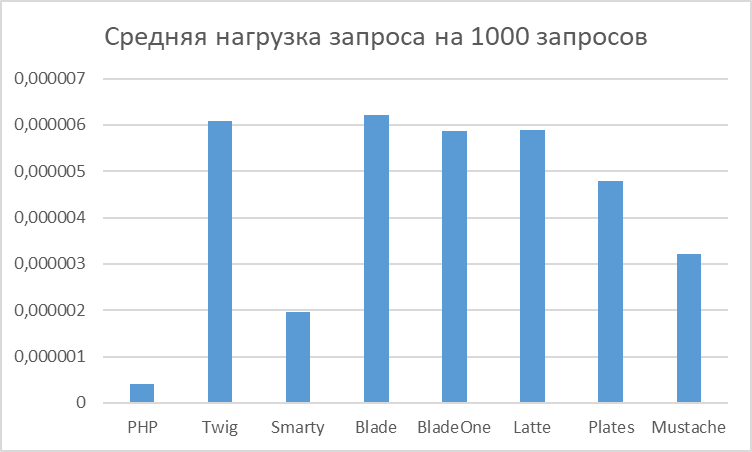
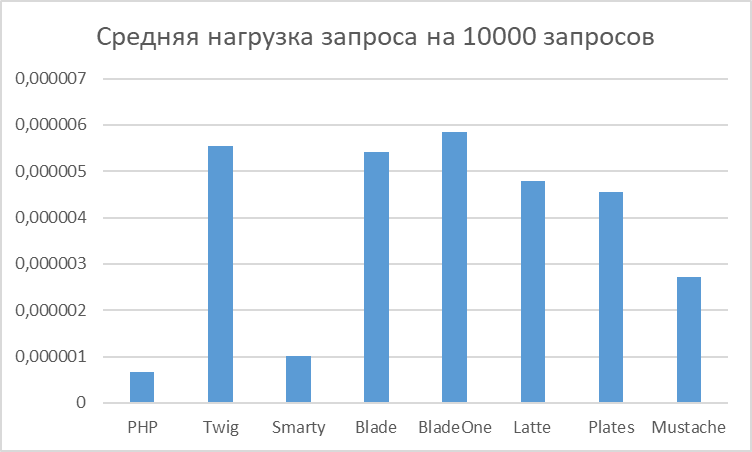
**Сравнение общей загрузки**

** **

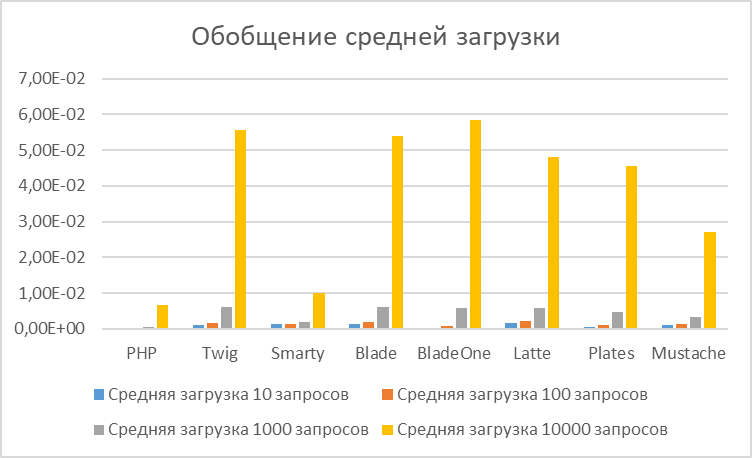
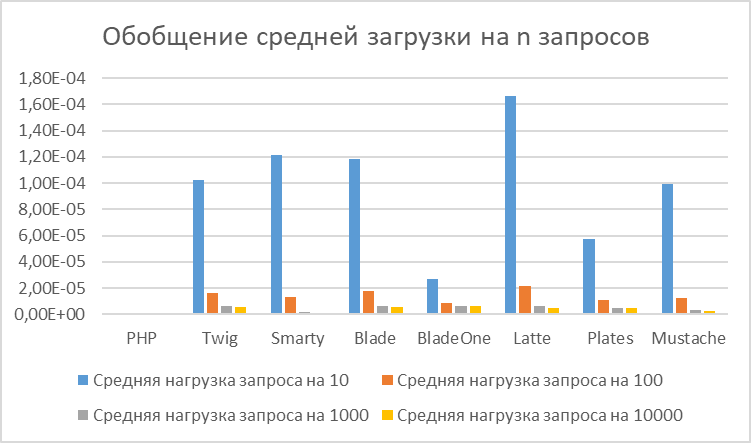
** **

**Сравнение средней нагрузки запроса**

** **

** **

**Обобщение общей нагрузки и средней нагрузки запроса**

** **

**ЗАКЛЮЧЕНИЕ И ОБСУЖДЕНИЕ**

Данная тема является актуальной и будет являться актуальной и через несколько лет, особенно для молодых разработчиков. Стоит учесть, что с каждым десятилетием будут появляться новые технологии, новые теории и новые методы, которые влияют на проектирование шаблонизаторов. В целом данная работа не устареет и через несколько лет, потому что здесь доступно и понятно изложены все этапы исследования, а значит повторить обзор шаблонизаторов не составит труда.

Практическая значимость исследования заключается в осознание минусов и плюсов той или иной технологии. А так как шаблонизатор является технологией, то точное знание скорости позволит подбирать программное обеспечение под проект. Вдобавок аналогичное исследование может провести статист, аналитик и разработчик в любом проекте, в котором появится необходимость изменить технологию или определить новый вектор развития.

Данная работа представляет особую важность для программистов-архитекторов или для архитектора программного обеспечения, так как ключевая обязанность архитектора — проектирование архитектуры ПО, то есть принятие ключевых проектных решений относительно внутреннего устройства программной системы и её технических интерфейсов. Эти точные исследования о скорости упростят задачу в выборе шаблонизатора для проекта.

Также в ходе исследования шаблонизаторы были разделены на три категории, каждая из которых в свое время делится на три вида. Виды в достаточном объеме описывают суть полученных результатов, а также их систематизируют и сравнивают. Графики в свою очередь наглядно подкрепляют полученный тезисы и могут и использоваться разработчиками, как краткие сводки о скорости шаблонизаторов.

Данное исследование имеет перспективы на продолжение. Так как оценивать шаблонизаторы можно и по другим характеристикам, а не только по скорости. Тем самым разработчики (и сам автор) будут владеть полной осведомленностью в данной теме. Единственная проблема – это актуальность. Со временем полученные данные перестанут быть актуальны, как и другие возможные исследования, а выводы – ошибочны.

Стоит сказать, что работа полностью развенчивает мифы по поводу скорости шаблонизаторов. Данный вопрос «Какой шаблонизатор на PHP является сама быстрым?» породил достаточное количество однотипных тем и без доказательственных споров. В заключение своей работы автор хочет сказать, чтобы люди помнили, что самое великое счастье для разработчика – это когда его технологией пользуются и, безусловно, кода есть вкусное кофе на столе.