

**МИНОБРНАУКИ РОССИИ**

**федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технологический университет «СТАНКИН»**

**(ФГБОУ ВО «МГТУ «СТАНКИН»)**

|  |  |
| --- | --- |
| **Институт**  **информационных**  **систем и технологий** | **Кафедра**  **информационных систем** |

|  |  |  |  |
| --- | --- | --- | --- |
| **Махмудов Бобурбек Нодирбекович** | | | |
| Тема: «**Исследование методов повышения производительности систем веб-шаблонов и разработка системы шаблонизаци на их основе**» | | | |
| **Выпускная квалификационная работа на присвоение квалификации «бакалавр» по направлению 09.03.02 «Информационные системы и технологии»** | | |
|  | Регистрационный № \_\_\_\_\_\_\_ | |
| **Заведующий кафедрой**  **д.т.н., проф.** |  | **Позднеев Б.М.** |
|  | подпись |  |
| **Руководитель**  **к.т.н., доцент** |  | **Бумарин Д.П.** |
|  | подпись |  |
| **Студент** |  | **Махмудов Б.Н.** |
|  | подпись |  |
|  |  |

ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 3](#_Toc34511621)

[ГЛАВА 1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 4](#_Toc34511622)

[1.1. Термины и определения 4](#_Toc34511623)

[1.2. Общий принцип работы систем веб-шаблонов 4](#_Toc34511624)

[1.3. Обзор существующих решений 8](#_Toc34511625)

[1.4. Основные причины низкой производительности шаблонизаторов 12](#_Toc34511626)

[1.5. Цели и задачи 12](#_Toc34511627)

[ГЛАВА 2. ИССЛЕДОВАНИЕ ВОЗМОЖНЫХ СПОСОБОВ ПОВЫШЕНИЯ ПРОИЗВОДИТЕЛЬНОСТИ СИСТЕМ ВЕБ ШАБЛОНОВ 13](#_Toc34511628)

[2.1. Устранение причины низкой производительности, связанной с использованием интерпретаторов 13](#_Toc34511629)

[2.2. Уменьшение системных вызовов 13](#_Toc34511630)

[2.3. Решение проблемы простоя системных ресурсов 13](#_Toc34511631)

[2.4. RESTful сервис, выполняющий роль системы веб шаблонов 13](#_Toc34511632)

[ГЛАВА 3. ПРОЕКТИРОВАНИЕ СЕРВИСА ПО ПРЕДОСТАВЛЕНИЮ УСЛУГ СИСТЕМЫ ВЕБ-ШАБЛОНОВ 14](#_Toc34511633)

[3.1. Моделирование работы сервиса 14](#_Toc34511634)

[3.2. Выбор инструментов 14](#_Toc34511635)

[3.3. Разработка сервиса 14](#_Toc34511636)

[3.4. Результат разработки, оценка производительности 14](#_Toc34511637)

[3.5. Потенциал к масштабированию 14](#_Toc34511638)

[ЗАКЛЮЧЕНИЕ 15](#_Toc34511639)

[СПИСОК ЛИТЕРАТУРЫ 16](#_Toc34511640)

[ПРИЛОЖЕНИЕ А. 17](#_Toc34511641)

[ПРИЛОЖЕНИЕ Б. 18](#_Toc34511642)

ВВЕДЕНИЕ

# АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

## Термины и определения

Система – комбинация взаимодействующих элементов, организованных для достижения одной или нескольких поставленных целей [1].

Веб-шаблон – HTML-код с готовым дизайном и версткой, а также с дополнительной разметкой на языке шаблонизации, который используется для построения веб-документа.

Шаблонизатор – программное обеспечение, позволяющее генерировать конечные веб-документы с использованием веб-шаблонов и с учётом контекста (данных).

Система веб-шаблонов (СВШ) – система, состоящая из трёх элементов:

1. веб-шаблонов;
2. источника данных (JSON, XML, база данных);
3. шаблонизатора.

Контекст шаблонизации – структура данных, содержащая переменные окружения и методы, которые могут быть использованы шаблонизатором в процессе построения веб-документа.

Абстрактное синтаксическое дерево (АСД) – конечное помеченное ориентированное дерево, внутренние вершины которого поставлены в соответствие (помечены) с операторами языка программирования, а листья — с операндами [3].

## Общий принцип работы систем веб-шаблонов

Все существующие системы веб-шаблонов функционируют похожим образом. Для работы системы необходимо выполнение следующих условий:

* наличие шаблонизатора – основной компонент системы, который генерирует конечный документ;
* наличием веб-шаблонов, размеченных языком шаблонизации, синтаксис которого поддерживается используемым шаблонизатором;
* наличие источника данных, при этом структура данных должна соответствовать правилам формирования контекста, который может быть обработан шаблонизатором.

При соблюдении вышеперечисленных условий, работу системы веб-шаблонов можно визуализировать на Рис. 1.2.1.



Рис. 1.2.1 Визуализация работы системы веб-шаблонов

Процесс шаблонизации в общем случае организован в последовательный набор операций, которые можно сгруппировать в четыре этапа:

1. Шаблон загружается в оперативную память. Источником шаблона может быть поток байт, файл, запись в базе данных и т.п.
2. Специальная подпрограмма, предназначенная для лексического анализа, именуемая «лексер», проводит анализ шаблона, и разбивает его на лексические единицы – лексемы, которые также называют токенами. Результатом этого процесса является поток токенов, которые значительно проще обрабатывать. Пример лексического разбора продемонстрирован на Рис. 1.2.1.
3. Другая подпрограмма, целью которой является синтаксический разбор потока токенов, именуемая «парсер», производит преобразование потока токенов в особую древовидную структуру данных, известной как абстрактное синтаксическое дерево.
4. На основе абстрактного синтаксического дерева и переданного контекста шаблонизации, шаблонизатор производит построение конечного веб-документа [3]. Пример построения абстрактного синтаксического дерева показан на Рис. 1.2.2.



Рис. 1.2.2 Лексический разбор шаблона, каждая лексема пронумерована



Рис. 1.2.3 Построение абстрактного синтаксического дерева

Далее можно ознакомится с функциональной моделью вышеописанного процесса в нотации IDEF0. На Рис. 1.2.3 представлена контекстная диаграмма процесса шаблонизации.



Рис. 1.2.4 Контекстная диаграмма процесса шаблонизации

На Рис. 1.2.4 приведена декомпозиция верхнего уровня процесса.



Рис. 1.2.5 Декомпозиция верхнего уровня процесса шаблонизации

## Обзор существующих решений

На сегодняшний день существует большое множество систем веб-шаблонов, как проприетарных, так и с открытым исходным кодом.

Системы веб шаблонов можно выделить в две основные категории, в зависимости от способа их размещения относительно конечного пользователя:

* server-side – работающие на стороне сервера;
* client-side – работающие в составе клиентского приложения.

Первые системы веб шаблонов работали на стороне сервера, и представляли из себя препроцессоры текста. Распространённой практикой было использование технологии CGI (Common Gateway Interface), которая подразумевает вызов веб-сервером внешнего программного кода, как правило скрипта, задачей которого была генерация веб-документа с использование данных, полученных в запросе от клиентского приложения. Со временем появились полноценные веб-фреймворки, которые имеют в своём составе систему веб-шаблонов в качестве отдельного программного модуля. Далее приведен перечень, наиболее популярных систем веб-шаблонов, работающих на стороне сервера.

Blade – входит в состав веб-фреймворка Laravel. Написан на языке PHP. Также как и сам фреймворк, частью которого является, Blade обладает свободной лицензией и открытым исходным кодом.

Django – по сути сам является веб-фреймворком, написанном на языке Python, обладает широкими возможностями шаблонизации. На основе синтаксиса языка шаблонизации, используемого в Django было создано большое количество других СВШ. Свободная лицензия.

Mustache – это скорее спецификация по созданию систем веб-шаблонов, нежели отдельно взятый программный продукт, но существует не мало имплементаций данной спецификации на разных языках программирования. Синтаксис, определенный в спецификации, является одним из самых распространённых и широко-используемых. Открытая спецификация [4].

Twig – система веб-шаблонов, синтаксис которой основан на Django. Написан на языке PHP, наиболее часто используется при проектировании веб-приложений по паттерну MVC (Model View Controller). Свободная лицензия.

Lasso – является сервером приложений, имеющий в своем составе одноимённую подсистему веб-шаблонов. Является примером проприетарной СВШ. По возможностям значительно уступает свободным аналогам.

Также существуют системы веб-шаблонов, которые могут исполнятся прямо в клиентском приложении, как правило это веб-браузер. Основной принцип работы этой категории СВШ состоит в том, что они взаимодействуют с объектной моделью документа [5] (DOM), с целью изменения его внешнего представления. Достигается это путём использования сценарных языков, наиболее популярным из которых является JavaScript. Ниже приведён список систем веб-шаблонов, исполняемых на стороне клиента.

Handlebars – синтаксис этой системы веб-шаблонов вдохновлён спецификацией Mustache, и имеет с ней полную обратную совместимость. Для ускорения генерации документа, Handlebars заблаговременно компилирует веб-шаблон в JavaScript код, что положительно сказывается на его производительности [6].

Squirrelly – легковесная встраиваемая система веб-шаблонов, написанная как библиотека на языке JavaScript, обладает высокими показателями быстродействия, эффективности. Не смотря на свой маленький размер, Squirrelly предоставляет широкий спектр возможностей шаблонизации.

Dot.js – по заявлению разработчиков является самым быстрым и лаконичным из всех существующих шаблонизаторов, написанных на JavaScript. Но из-за своей легковесности, не предоставляет всех функциональных возможностей аналогов.

Все приведённые выше системы веб-шаблонов были разработаны для решения определённых задач, каждый из них обладает рядом достоинств и недостатков. Выбор той или иной системы веб-шаблонов, да и любого программного средства в целом, как правило обусловлен несколькими факторами, среди которых можно выделить:

* Уровень навыков и опыты команды разработчиков в применении определённой технологии. Разработчики часто отдают свои предпочтения уже известным и проверенным решениям, так как внедрение новой технологии всегда подразумевает определённые риски, связанные с неопределённостью.
* Окружение. А именно какой перечень технологий уже используется, и как хорошо новое программное средство будет взаимодействовать с существующим окружением.
* Удобство в использовании. Насколько продукт понимаемый, изучаемый, используемый и привлекательный для [пользователя](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C) в заданных условиях [2]. Иначе говоря, пользователь должен иметь возможность эксплуатировать программное средство в определенных условиях для достижения установленных целей с необходимой результативностью, эффективностью и удовлетворённостью.
* Удобство в сопровождении. Сколько усилий нужно прилагать для доработок и поддержки программного продукта в соответствии с изменяющимися требования заказчиков.
* Производительность. Насколько быстро работает программное обеспечение под определённой нагрузкой. Часто пренебрегаемый критерий выбора, так как на практике его перевешивают предыдущие пункты. Это связано с тем что быстродействие системы веб-шаблонов не является «узким местом» для большинства надсистем в которой она функционирует, и время, затрачиваемое на выполнение ею полезной работы, находится в «приемлемых» пределах.

Именно производительность и эффективность, и являются центральной темой данной выпускной квалификационной работы.

## Основные причины низкой производительности шаблонизаторов

## Цели и задачи

# ИССЛЕДОВАНИЕ ВОЗМОЖНЫХ СПОСОБОВ ПОВЫШЕНИЯ ПРОИЗВОДИТЕЛЬНОСТИ СИСТЕМ ВЕБ ШАБЛОНОВ

## Устранение причины низкой производительности, связанной с использованием интерпретаторов

## Уменьшение системных вызовов

## Решение проблемы простоя системных ресурсов

## RESTful сервис, выполняющий роль системы веб шаблонов

# ПРОЕКТИРОВАНИЕ СЕРВИСА ПО ПРЕДОСТАВЛЕНИЮ УСЛУГ СИСТЕМЫ ВЕБ-ШАБЛОНОВ

## Моделирование работы сервиса

## Выбор инструментов

## Разработка сервиса

## Результат разработки, оценка производительности

## Потенциал к масштабированию

ЗАКЛЮЧЕНИЕ

СПИСОК ЛИТЕРАТУРЫ

1. ГОСТ Р ИСО/МЭК 15288-2005. Информационная технология. Системная инженерия. Процессы жизненного цикла систем.
2. ГОСТ Р ИСО/МЭК 25010-2015 Информационные технологии (ИТ). Системная и программная инженерия. Требования и оценка качества систем и программного обеспечения (SQuaRE). Модели качества систем и программных продуктов.
3. Михаил Вюрш. Улучшение распознавания изменений в исходных кодах с помощью абстрактных синтаксических деревьев: дипломная работа [Текст]. Университет Цюриха 2006. – 64 с.
4. Twig Internals – Documentation [Электронный ресурс]. Режим доступа: <https://twig.symfony.com/doc/2.x/internals.html>. (Дата обращения: 07.03.2020).
5. Chris Wanstrath. Mustache – Logic-less templates [Электронный ресурс]: 2009. Режим доступа: <https://mustache.github.io/mustache.5.html>, свободный. (Дата обращения: 07.03.2020).
6. Flanagan, David. JavaScript: The Definitive Guide [Текст]. O'Reilly & Associates, 2006. – 992 с.
7. Introduction | Handlebars [Электронный ресурс]. Режим доступа: <https://handlebarsjs.com/guide/#what-is-handlebars>, свободный. (Дата обращения: 07.03.2020).

ПРИЛОЖЕНИЕ А.

**ДОКУМЕНТАЦИЯ API**

ПРИЛОЖЕНИЕ Б.

**СИНТАКСИС ДЛЯ НАПИСАНИЯ ШАБЛОНОВ**