**РЕЧЬ**

**Слайд #2**

Данная презентация структурирована следующим образом:

* Сначала я в краткой форме ознакомлю вас с предметной областью, с актуальностью, выполненной мной работы, а также с целями и задачами, которые преследовались в ходе её выполнения.
* Далее будет рассмотрен перечень основных причин низкой производительности систем веб-шаблонов, после чего будут приведены найденные мной решения для их устранения.
* Остальная часть презентации содержит описание проекта разработанной системы, выбранных технологий реализации и результатов тестирования.

**Слайд #3**

Но для начала небольшая вводная часть про системы веб-шаблонов.

Собственно, система веб-шаблонов (template engine на англ.) – это система, состоящая из трех компонентов:

1. Веб-шаблон (некий макет веб-документа).
2. Источник данных (данные которыми нужно заполнить веб-документ) – это может быть любой объект, который содержит данные (БД, XML, JSON).
3. Шаблонизатор – специальное программное обеспечение для комбинирования веб-шаблона и источника данных согласно установленному набору правил.

Иначе говоря, системы веб-шаблонов предназначены для автоматизированной массовой генерации веб-документов.

**Слайд #4**

На данной диаграмме IDEF0 представлен процесс построения веб-документа системой веб-шаблонов.

**Слайд #5**

Актуальность данной работы заключается в том, что существующие решения не эффективны в плане потребления вычислительных ресурсов, также большинство из них предназначены для использования в составе более крупных веб-фреймворков, что делает их не совсем удобными в эксплуатации людьми не технического профиля. То есть разработав производительную систему веб-шаблонов с высокими показателями удобства эксплуатации позволит экономить как вычислительные, так и трудовые ресурсы.

**Слайд #6**

Как было сказано ранее, цель данной работы — это поиск способов повышения производительности веб-шаблонов.

**Слайд #7**

На этом слайде представлены основные факторы, которые негативно сказываются на производительности системы веб-шаблонов:

1. Подавляющее большинство существующих решений написаны на интерпретируемых языках (PHP, Python, JavaScript).
2. Присутствие механизм автоматического управления памятью, также известного как сборщик мусора.
3. Не всегда присутствует кеширования обработанных шаблонов, что приводит к необходимости повторной обработки.
4. Вычислительные ресурсы не используются полностью, из-за исполнения программы в одном потоке, и наличием блокировок, связанных с чтением и записью с медленных хранилищ данных.

**Слайд #8**

Рассмотрим, как интерпретируемые языки отличаются от компилируемых.

При компиляции, компилятор обрабатывает исходный код и производит оптимизированных машинный бинарный код под целевую архитектуру ЭВМ.

При интерпретации обработка исходного кода происходит построчно, и результатом такого процесса является промежуточный код виртуальной машины, так же известной как байт-код. Далее этот байт-код подвергается процессу трансляции в машинный код целевой архитектуры.

Такой подход в несколько раз медленнее по сравнению с процессом компиляции.

**Слайд #9**

Простой вычислительных ресурсов подразумевает, что в многоядерных процессорах при работе системы веб-шаблонов используется лишь одно ядро, в то время как остальные бездействуют.

**Слайд #10**

Для решения ранее описанных проблем, были применены следующие подходы:

1. Вместо интерпретируемого языка необходимо использовать компилируемый язык программирования.
2. В выбранном языке программирования не должно быть механизма сборки мусора, а для упрощения управления памятью необходимо применять парадигму RAII (объект который получил память, ответственен за её освобождение).
3. В качестве кеширования использовать алгоритм LRU (вытеснение наименее используемых объектов).
4. Реализация системы веб-шаблонов как веб-сервис, который может параллельно и асинхронно обрабатывать большое количество запросов на построение веб-документов.

**Слайд #11**

Немного слов о веб-сервисе, данный веб-сервис является RESTFul, то есть он имеет клиент-серверную архитектуру, и взаимодействие с клиентами происходит посредством стандартных HTTP запросов.

**Слайд #12**

Укрупнённо модель веб-сервиса выглядит следующим образом:

Множество клиентов делает обращения к сервису посредством HTTP запросов, сервис перенаправляет запросы на соответствующие модули в зависимости от запроса.

**Слайд #13**

На данном слайде приведена IDEF0 диаграмма обработки таких HTTP запросов, по ней видно, что после предварительной обработки запроса веб-сервис перенаправляет запрос на соответствующую подсистему для дальнейшей обработки.

**Слайд #14**

Для реализации ранее описанного веб-сервиса были использовано следующие технологии:

Серверная часть была написана на языке программирования Rust. Для поддержки многопоточного и асинхронного исполнения была использована библиотека tokio. А для непосредственно самой шаблонизации в веб-сервис был интегрирован шаблонизатор tera.

Пользовательский веб-интерфейс был написан с применением javascript-фреймворка vuejs и использованием стандартных html и css

**Слайд #15**

На этом слайде приведён пример пользовательского веб-интерфейса разработанного сервиса.

**Слайд #16**

По завершению разработки были проведены тесты для оценки производительности.

Тестирование велось на виртуальном сервере от Google, на слайде приведены характеристики этого сервера.

На диаграмме показано соотношение одновременных подключений в конкретные моменты времени и время ответа веб-сервиса. А именно на 550 одновременных подключений средне время ответа составляет чуть более 4ёх секунд.

**Слайд #17**

На данной картинке можно ознакомиться с нагрузкой, которая фиксировалась на сервере во время проведения теста. По ней видно, что оба ядра процессора нагружены на 50%, т.е., не смотря на большое количество запросов, разработанная система веб-шаблонов хорошо справляется с поставленными задачами.

**Слайд #18**

В заключении можно сказать что все поставленные цели были достигнуты: были проанализированы проблемы низкой производительности веб-шаблонов, предложены способы устранения данных проблем, после чего разработана высокопроизводительная и удобная в использовании система веб-шаблонов.

**Слайд #19**

СПАСИБО ЗА ВАШЕ ВНИМАНИЕ