**Слайд 1(титул)**

Здравствуйте, меня зовут Тактаров Евгений и тема моей работы:

Разработка системы проведения опросов аудитории во время публичных выступлений

**Слайд 3(Постановка задачи)**

Моя задача состояла в создании веб-сервиса в котором можно:

* Создавать
* Проводить

Опросы. Во время выступлений в опросах участвуют с мобильных устройств, а результаты сразу же отображаются на главном экране.

**Слайд 4(Инструменты)**

* Для решения этой задачи я использовал Node.js и фреймворк Express как веб-сервер.
* Vue.js для создания интерфейса пользователя
* Websocket’ы для обеспечения двусторонней связи клиента и сервера

**Слайд 5(диаграмма-структура)**

На слайде изображена структура веб-сервиса:

* Express обрабатывает HTTPзапросы и передает их в модель данных
* Vue – реактивный фреймворк. Когда меняются данные то он автоматически перестраивает страницу

**Слайд 6(Express)**

Express не сложен в использовании но крайне гибок:

* Основан на написании обработчиков путей HTTP запросов
* Пути можно описывать регулярными выражениями
* Обработчики можно выстраивать в цепочки
* Дополнения добавляют функционал

**Слайд 7(Требования к модели данных)**

Чтобы создать модель данных, я прежде всего выявил список требований:

* Параллельность опросов
* Наличие коротких ссылок
* Защита опросов
* Масштабируемость

**Слайд 8(модель)**

В результате в приложении модель представлена через JS-объект

* Он рекурсивно состоит из списков, словарей и объектов со свойстами и методами,инкапсулирующие взаимодействия с моделью
* Каждый пользователь это объект его WS, события которого вызывают методы модели
* Объект инициализируется и экпортируется из отдельного JS модуля, что имитирует шаблон одиночки.

**Слайд 9(WS)**

В современных веб-браузерах вебсокеты уже встроены, а на сервере я использую расширение для Express

Я реализовал через WS простой JSON протокол, в котором если клиент дает некорректные данные, то сервер закрывает соединение

**Слайд 10(WS диаграмма)**

На этом слайде описан алгоритм подключения нового клиента через WS:

* Express обрабатывает только начало соединения и затем передает его в модель
* Первое сообщение от клиента всегда содержит все его «паспортные» данные
* Промедление первого сообщения тоже влечет за собой закрытие

**Слайд 11(Проблема)**

Одна из интересных и важных проблема с которой я столкнулся –

Переполнение вызовов

Когда один клиент отправляет данные и изменяет модель, то нужно обновить данные у всех клиентов.

Проще всего вызывать функцию «отправить всем» в конце обработчика сообщения от клиента. Но особенно в однопоточном Node.js когда много клиентов часто вносят изменения, то на них посыплется лавина из уже неактуальных данных.

**Слайд 12(Debounce)**

Для решения этой проблемы я использую обертку для функции Debounce. Она откладывает исполнение через промежуток от последнего.

В примере на слайде происходит исполнение функции при первом вызове, несколько отложенных в очередь вызовов, исполнение функции и очистка очереди.

Обернув функцию «отправить всем», нескольким изменениям модели будет соответствовать только одно обновление клиентов.

**Слайд 13(Устройство клиентской части)**

На этом слайде представлено устройство клиентской части:

* Когда клиент запрашивает страницу участия в опросе - сервер собирает из шаблона основу страницы с «паспортными» данными
* В браузере страница подгружает фреймворк из CDN, а код и стили из статичных файлов сервера
* Vue связывает данные и открывает Websocket соединение
* По приходу данных страница перерисовывает компоненты

**Слайд 14(Скрины)**

Так выглядит интерфейс пользователя.

**Слайд 15(Результаты)**

Полученные в работе результаты представлены на слайде