

Почему мы сделали свои A/B тесты

Мы пробовали чужое

По итогам внутрикорпоративного отбора победил Optimizely, который мы и пытались использовать

проблемы

jQuery

внешний CDN

JS-совместимость с rich JS сайтами — надо много что писать руками

Прыгающий gender страницы

Языки — их нет, надо выкручиваться

Замены с переменными (пол, склонения) — та же самая беда

Выбор пользователей по каким-то переменным типа страны, пола — все надо делать в PHP-коде

Очень ограниченное применение — нет API, с мобильным приложением уже не поэкспериментируешь

Но, главное, они вообще не то анализируют что нужно

Большинство докладов про A/B тестирование — из мира e-commerce, где все как-то быстро и просто. Приходит пользователь — анализируем три страницы — главную, страницу каталога, страницу товара — далее воронка продаж — и оп, у нас простая цель — покупка, и... следующий!

Мы — не интернет магазин, где каждый пользователь покупает один раз и уходит

Нам нужна статистика по большому количеству параметров — и с большим сроком, чем от момента захода на сайт до покупки

Мы хотим видеть эффект от новых фишек в изменившемся поведении пользователей (долгосрочность)

Мы хотим видеть влияние тестов на какой-то смежный функционал сайта, например увеличение кол-ва чатов, продолжительности посещения

Чем свое лучше и хуже

Плюсы

Гораздо большая применимость — WAP-сайт, мобильное приложение, количественно-временной функционал (типа времени отправки писем)

Выдерживает большие нагрузки

Никакого дурацкого JS с внешнего CDN с прыгающими страницами, только PHP, только хардкор

Весь нужный вам функционал типа поддержки языков и склонений теперь доступен

Минусы

Придется проектировать и кодить свою систему

Придется поразбираться в теории вероятности

Нагрузка теперь ваша — нужно место на дисках, процессорное время

Статистику теперь тоже считаем сами — коддинг, место, нагрузки

Сделать свой WYSIWYG-интерфейс редактирования сложно (но и не нужно)



Как бы нам все это хотелось чтобы работало с высоты птичьего полета

Подготавливаем набор шаблонов

или поведений, или констант, которые мы будем тестировать

Задаем критерии отбора пользователей для участия в тесте

Привязываем определенный шаблон/поведение/значение переменной к каждому юзеру, участвующему в тесте

Выбираем метрику, по которой мы будем оценивать результат

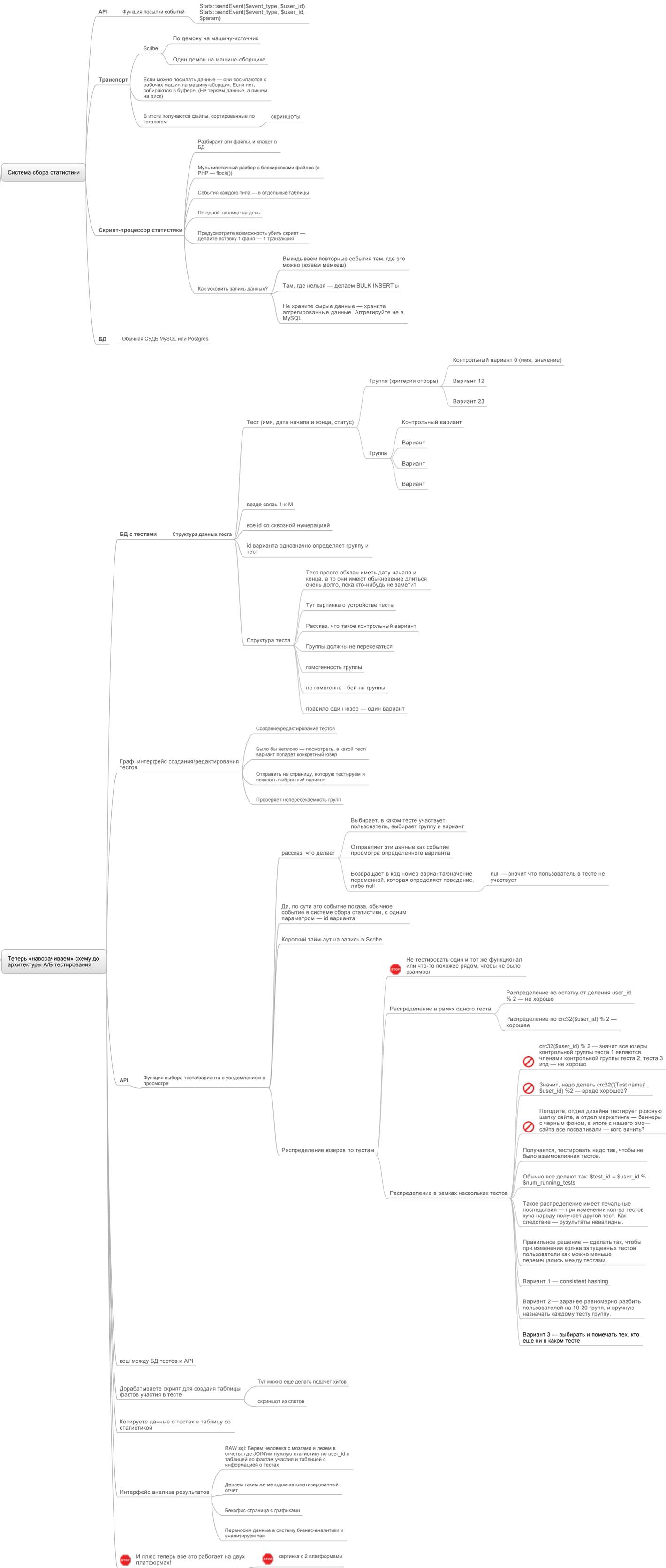
Делаем так, чтобы у нас была статистика по этой метрике с привязкой к user_id

Собираем информацию о том, какому пользователю какой шаблон показывали

Запускаем тест — завершаем тест

Берем кого-то умного и анализируем результаты

Архитектура нашего решения



Как теперь все это дело работает

Распикиваете по коду статистику

Обработчик веб-запроса

дергает

Функция выбора теста/варианта, она же сама пишет в статистику о просмотре варианта

показывает

Подготовленный шаблон с другим дизайном/другое поведение

В итоге у нас получается БД где есть таблицы с фактами действий и есть таблица участия в тесте. А также таблицы описывающие тесты

Анализируем результаты

