Переписать проект и выжить

Андрей Брюханов

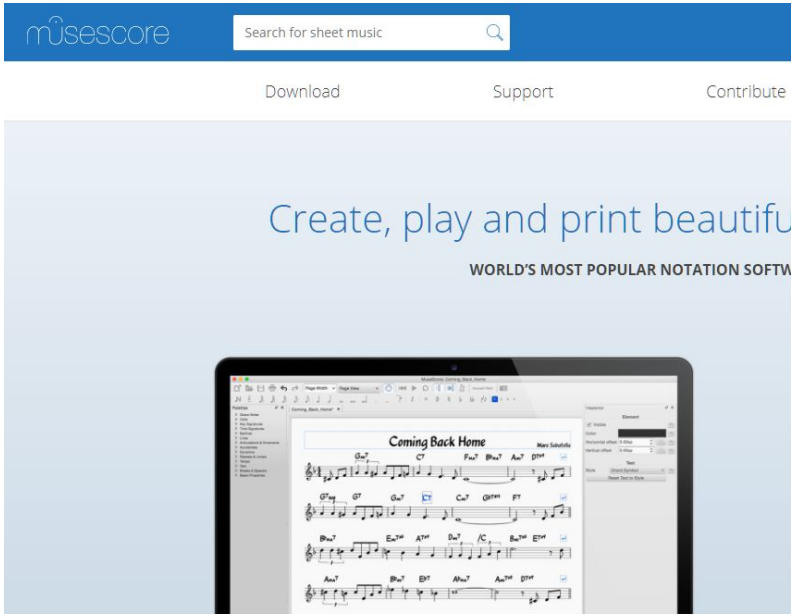
http://www.devconf.ru

Это я:

- 10+ лет опыта
- фрилансер, джуниор, тимлид и пр.
- разработчик в ultimate-guitar.com
- разработчик musescore.com
- всегда не прочь поковырять legacy

Это Musescore:

- крупнейшая библиотека партитур
- самый популярный нотный редактор
- 1.2 млн. партитур
- 5.5 млн. пользователей
- с осени 2017 - часть UltimateGuitar

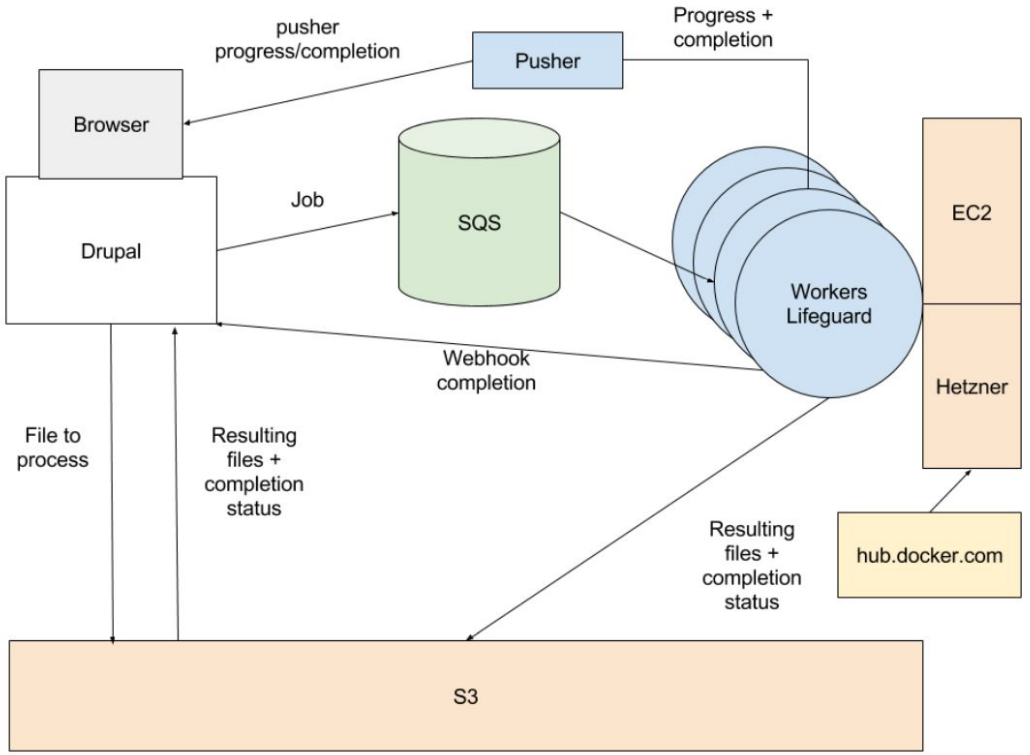


История:

- создан 6 или 7 лет назад
- создан энтузиастами

Изнанка проекта:

- Drupal
- shared hosting Acquia
- Amazon AWS
- C++, Java, Docker
- CloudFront, Fastly
- Heroku
- ...



ЭЭЭ

Цели:

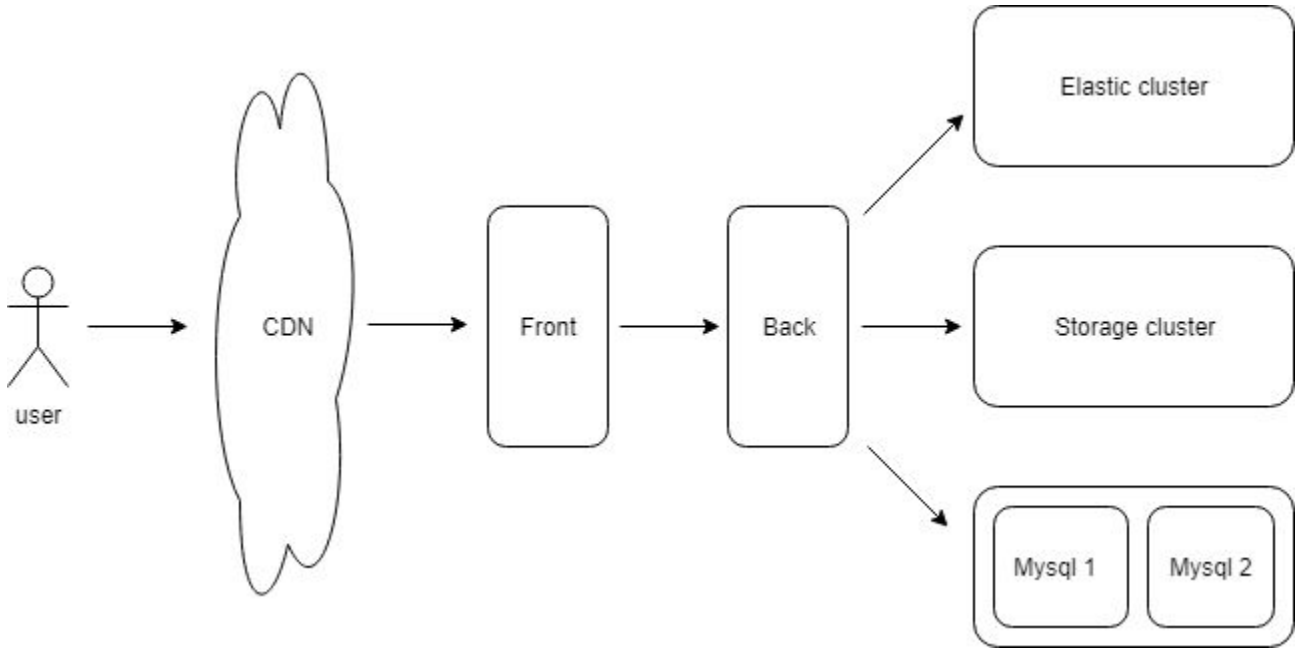
- проект должен приносить доход
- максимальная простота и удобство разработки
- полный контроль над окружением

Цель 1, доход:

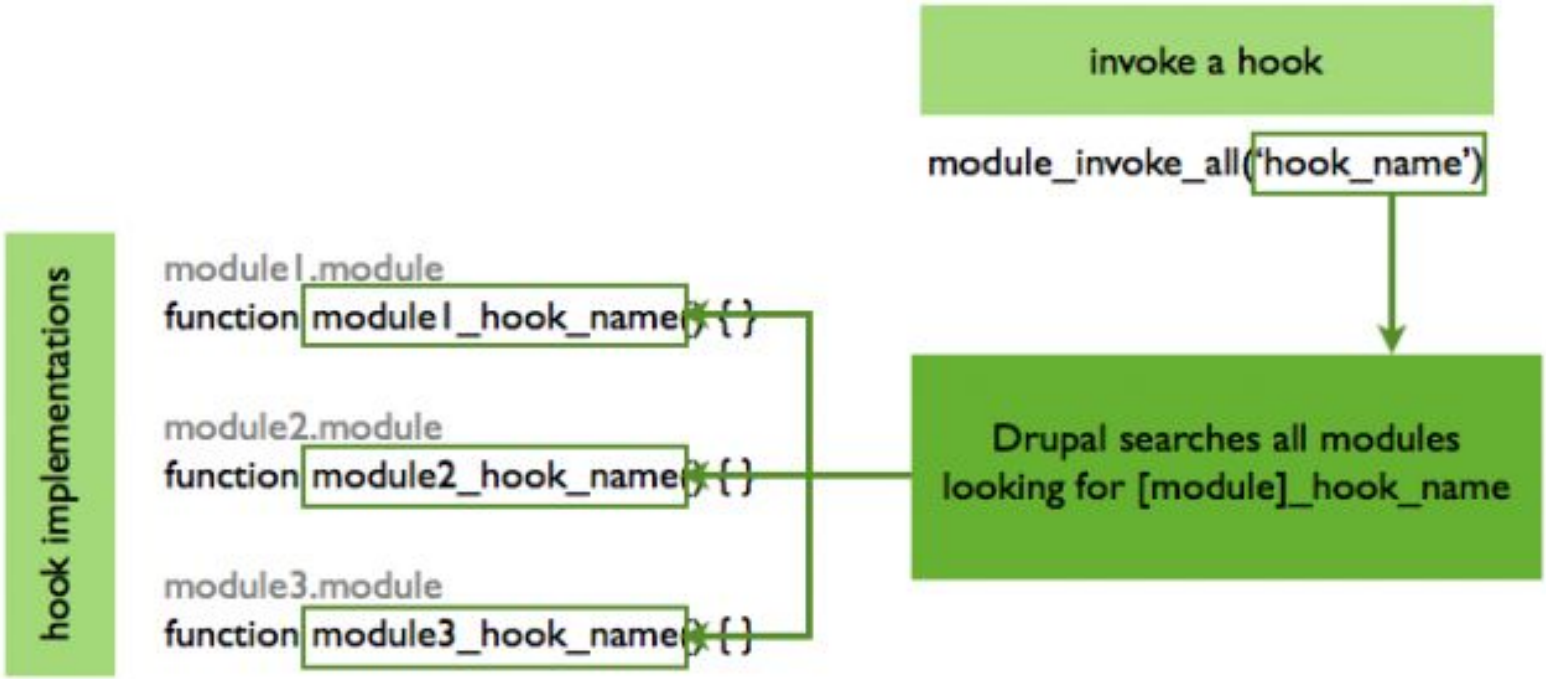
- отказ от лишних облачных сервисов
- сокращение нагрузки на сервера
- монетизация, реклама etc.

Цель 2, простота разработки:

Цель 3, контроль над окружением:



Производительность D8, хуки:



Производительность D8, БД:

- отдельная таблица на каждое поле форм
- большое количество накладных расходов

Производительность D8, кеширование:

- кеш данных - D8
- кеш блоков - D8
- кеш страниц - CDN

Все переписать!

(на самом деле нет)

Правила игры:

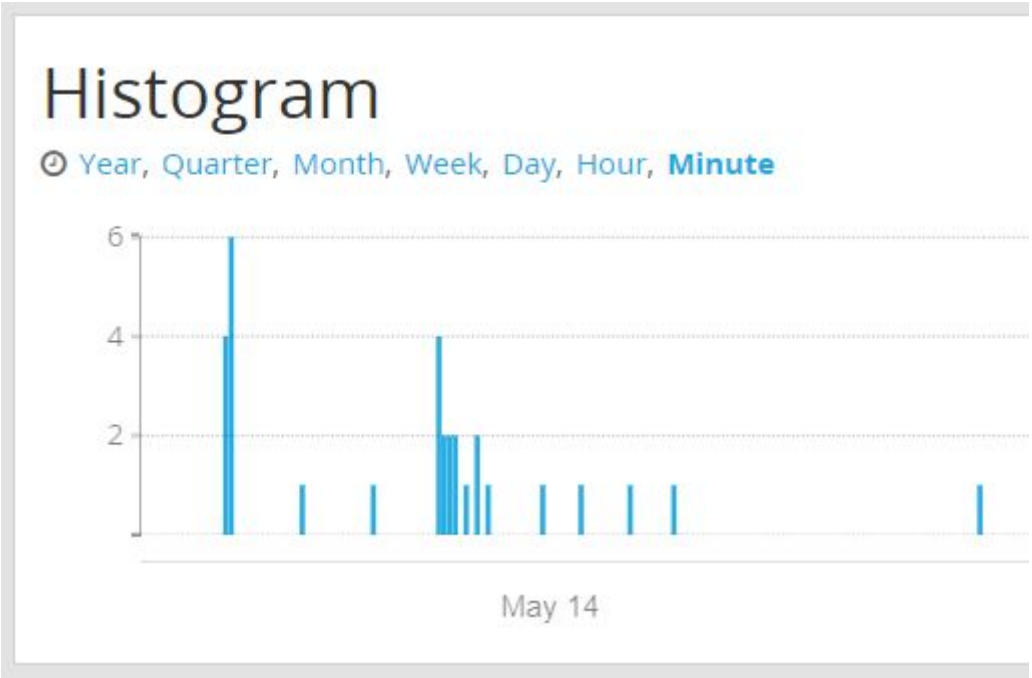
- непрерывная работа сервиса
- пользователи не должны страдать
- допустимы временные костыли и велосипеды

С чего начать:

- переезд на свой сервер
- человеческий деплой
- тотальный мониторинг

Что мониторить:

- состояние сервера
- логи и ошибки
- финансовые метрики
- пользовательские метрики



Что переносить сначала:

- связи и зависимости модулей
- создаваемая нагрузка
- планы на развитие
- необходимость изменения БД

Фундамент:

- роутинг
- авторизация
- работа с БД
- Yii2 - D8 bridge
- D8 - Yii2 bridge

Роутинг:

- D8 поддерживает произвольные алиасы и в проекте они активно используются
- частные виды путей разруливает nginx
  - /group/123
  - /user/123/scores/124
  - /user/1
- общие виды с определенного момента роутятся на Yii и если тот возвращает 404, то уходят на D8
  - /node/111
  - /thomas/score1

Авторизация:

- авторизация строго через D8
- разные бекэнды, разные сервера, использовать общую сессию невозможно
- D8 хранит специальную куку и id сессии в БД
- при каждом запросе читаем куку и проверяем наличие записи в БД - дополнительный запрос, но удобство пользователя
- пришлось реализовать корректную генерацию хешей как в D8

Хеширование:

```php
public static function hashBase64($data)
{
    $hash = base64_encode(hash('sha256', $data, TRUE));
    // Modify the hash so it's safe to use in URLs.
    return str_replace(['+', '/', '='], ['-', '_', ''], $hash);
}
```

БД:

- Drupal way - бессмысленный и беспощадный: любая запись (нода) состоит из полей, каждое поле хранится в отдельной таблице
- каждая таблица содержит массу избыточной информации
- читать не проблема, сложность с записью
- реализовали поведения для моделей: генерация UUID, автогенерация delta поля
- составные primary key

Типичная табличка:

bundle	deleted	entity_id	revision_id	langcode	delta	roles_target_id
user	0	1	1	en	0	developer
user	0	1	1	en	1	pro
user	0	1	1	en	2	administrator
user	0	1	1	en	3	staff
user	0	1	1	en	4	moderator

```php
private function shiftDelta()
{
    $sql = 'UPDATE `' . self::tableName() . '`
        SET `delta` = `delta` + 1
        WHERE `deleted` = :deleted
            AND `entity_id` = :entity_id
            AND `langcode` = :langcode
        ORDER BY `delta` DESC';
    self::getDb()
        ->createCommand($sql, [
            ':deleted'   => 0,
            ':entity_id' => $this->entity_id,
            ':langcode'  => self::DEFAULT_LANGCODE,
        ])
        ->execute();
}
```

Работа со связями БД:

- все связи только через геттеры
- пара геттеров на каждую связь:
  - один определяет зависимую модель
  - второй непосредственно получение значения
- при переписывании просто выкидывается геттер, определяющий связь и немного правится геттер, получающий значение

```php
public function getSubtitle(): string
{
    return $this->relationSubtitle ? $this->relationSubtitle->field_score_subtitle_value : '';
}
public function getRelationSubtitle()
{
    return $this->hasOne(Relation\ScoreSubtitle::class, ['entity_id' => 'nid']);
}
```

```
public function getRelationPartCount() {...}
public function getRelationLength() {...}
public function getRelationPageCount() {...}
public function getRelati
public function getRelati
public function getRelati
public function getRelati
public function getRelati
public function getRelati
public function getRelati
```

```
public function getRelationLicense() {...}
public function getRelationComposer() {...}
public function getRelat
public function getRelat
public function getRelat
public function getRelat
public function getRelat
public function getRelat
public function getRelat
```

```
public function getRelationScoreLanguage() {...}
public function getRelationScoreHasCustomAudio() {...}
public function getRelationSubtitle() {...}
public function getRelationScorePoet() {...}
public function getRelationScoreLyricsIn() {...}
public function getRelationScoreChordsIn() {...}
public function getRelationScoreDimensions() {...}
public function getRelationScorePartPrograms() {...}
public function getRelationScorePartNames() {...}
public function getRelationScoreFile() {...}
public function getRelationScoreBraille() {...}
public function getRelationMedia() {...}
public function getRelationScoreClaimer() {...}
```

Yii2 D8 bridge:

- необходим для выполнения действий, которые нельзя решить простой записью в БД, например, сброс кеша
- нет времени (и желания) изучать API D8, писать к нему клиента и вообще
- простой контроллер с авторизацией по ключу
- простые GET запросы, малое количество действительно необходимых действий

D8 Yii2 bridge:

- любая передача данных должны идти только через БД
- частичное исключение - сбор статистики по действиям пользователей через AJAX запросы
  - нажатие play/pause в плеере
  - печать партитуры и тд

Миграция логики:

- D8 строится на хуках, любое действие может быть изменено через любой модуль, явные вызовы отсутствуют
- комбинированный подход
  - честный анализ
  - работа с черным ящиком

Черный ящик:

- список изменяемых параметров
  - авторизован или нет
  - роль пользователя на сайте
  - владелец материала или нет
  - ...
- оптимально для переноса компонентов страниц
  - блоки
  - меню

Сопутствующие потери:

- сломали поиск в мобильном API
- потеряли некоторые email уведомления
- разные мелкие баги

Мнение пользователей:

- недовольство старожилов
- многочисленные мелкие баги и постоянные доработки
- сложность продажи подобных изменений пользователям

Очистка кармы:

- общение
- оперативная обратная связь
- видимые улучшения

Итог:

- полгода, 80% функционала старого сайта
- ежемесячные затраты снижены вдвое
- сохранено сообщество и динамика роста