Белгородский Государственный Технологический Университет им. В. Г. Шухова

Кафедра программного обеспечения вычислительной техники  
и автоматизированных систем

Курсовая работа

по дисциплине «Конструирование программного обеспечения»  
  
тема: «Разработка сервиса для стриминга изображений»

**Автор работы:**  
студент группы ПВ-41  
Адаменко И. И.

**Руководитель работы:**  
Паньков С. В.

Белгород  
2015

Оглавление

[Задание и цель курсовой работы 3](#_Toc439045770)

[Введение 4](#_Toc439045771)

[Основные понятия 5](#_Toc439045772)

[Инструментарий 7](#_Toc439045773)

[Клиентская часть 8](#_Toc439045774)

[Рабочий процесс 8](#_Toc439045775)

[Инструменты 9](#_Toc439045776)

[Реализация 10](#_Toc439045777)

[Заключение 11](#_Toc439045778)

# Задание и цель курсовой работы

Разработать веб-сервис, позволяющий пользователям хранить и делиться изображениями друг с другом в онлайн-режиме, тегировать их и сортировать по стримам.

# Введение

В наше время в интернете огромное количество сервисов, которые предоставляют пользователям возможность хранить у себя изображения и другие файлы, однако, сервисов, которые помогали бы делиться этими изображениями с другими пользователями не так много. Речь не о социальных сетях и больших медиа-хостингах, а о самостоятельных ресурсах, главной целью которых является хранение и обмен изображениями между пользователями.

Дабы разрешить эту проблему целью этого курсового проекта является написание именного такого веб-сервиса. Актуальность этой проблемы также обусловлена развитием облачных технологий и отказом пользователей от хранения медиа-контента у себя на жёстком диске. Разработав этот сервис, мы поможем им избавиться от занимаемого данными объёма, а также предоставим возможность находить новые, интересующие пользователей изображения и управлять старыми с помощью тегирования и создания стримов.

# Основные понятия

Ключевым понятием, лежащим в основе приложения, является *стрим*. Под стримом понимается поток изображений, объединённых пользователем в единое целое.

Стримы создаются пользователями как некие выходные каналы данных. Они могут наполнять их как своим контентом, так и чужим. В этом особенность стрима — их можно смешивать друг с другом с помощью тегов. Обычный сценарий создания стрима выглядит следующим образом:

1. Пользователь решает создать стрим с автомобилями и мотоциклами, но у него есть только фотографии мотоциклов.
2. Пользователь находит стримы с автомобилями и помечает их как источники данных.
3. Поскольку ему не нравятся российские автомобили, он создаёт фильтр для этих стримов и добавляет туда тег «российские автомобили».
4. Теперь у пользователей есть стрим мотоциклов и автомобилей, однако в нём нет тех, которые произведены в России.

Вторым по важности ключевым понятием является *тег.* Тег — это метка, слово или словосочетание, кратко и ёмко характеризующее стрим. Один стрим может быть описан нескольким количеством тегов, причём, теги не должны быть уникальными, в этом их суть. То есть, один может (и должен, только тогда от него будет польза) быть привязан к нескольким стримам.

Структура тегов древовидна, причём, каждый тег-ребёнок характеризует подмножество своего тега-родителя. Таким образом пользователь может конкретизировать то, что хочет добавить или исключить. В примере выше пользователь мог видеть перед собой, например, такую структуру (жирным помечен его выбор):

* ...
* Игры
* Интерьер
* Космос
* **Машины**
  + Авиация
  + **Автомобили**
    - …
    - Немецкие
    - **Российские**
    - Французские
    - ...
  + Мотоциклы
  + Тяжёлая техника
* Минимализм
* Мужчины
* Оружие
* …

Пользователи могут создавать свои теги и помечать их место в общей структуре тегов. Позже пользователи с бóльшими правами (модераторы) могут изменить теги и поменять их место в структуре.

Помимо тегов и стримов у каждого пользователя есть свои настройки, благодаря которым фильтруется получаемый им контент. Например, переключатель «Показывать 18+», который поможет отсеять изображения для взрослой аудитории.

За каждым пользователем также закреплена страница со списком его стримов, на которую может зайти любой другой пользователь, выбрать понравившиеся и включить их в какие-то свои стримы.

# Инструментарий

Вся разработка ведётся двумя программистами, первый из которых пишет клиентскую часть, второй — серверную. Для синхронизации рабочего процесса используется распределённая система контроля версий Mercurial, с веб-интерфейсом в сервисе Bitbucket. Это позволяет каждому из разработчиков своевременно получать уведомления о том, какие изменения внесены в код, где именно и почему.

Для разработки клиентской части и для её работы используются следующие технологии:

* Node.js — программная платформа, с помощью которой происходит компиляция исходных файлов клиента и сборка их в необходимую структуру для работы с сервером.
* Gulp.js — потоковый сборщик проектов на JavaScript.
* Sass (SCSS) — метаязык на основе CSS, предназначенный для увеличения уровня абстракции CSS-кода и упрощения файлов каскадных таблиц стилей; транслируется в CSS во время сборки.
* Semantic UI — фреймворк для создания переносимых интерфейсов, который позволяет повторно использовать элементы UI из их общей библиотеки; используется для прототипирования.
* jQuery — библиотека JavaScript, фокусирующаяся на взаимодействии JavaScript и HTML и помогающая легко получать доступ к любому элементу DOM.

Для разработки серверной части же используются следующие технологии:

* Tornado — веб-сервер и фреймворк, написанный на Python.
* MongoDB — документо-ориентированная система управления базами данных.

# Клиентская часть

## Рабочий процесс

Основной рабочий процесс над клиентской частью заключается в написании (вёрстке) HTML-шаблонов для отображения данных, поступающих с сервера. Для того, чтобы структурировать шаблоны и облегчить их разработку и поддержку, используется БЭМ.

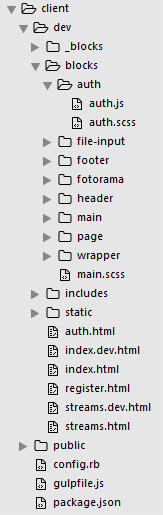
БЭМ — это методология разработки веб-приложений, которая заключается в разделении всех шаблонов на такие сущности как блок, элемент и модификатор (отсюда и название).

Блок — это логически и функционально независимый компонент страницы, инкапсулирующий в себе поведение (JavaScript), шаблоны (HTML), стили (CSS) и другие технологии реализации (например, PNG).

Элемент — составная часть блока, которая не может использоваться в отрыве от него. Например, пункт меню вне контекста блока меню не используется, а значит он является элементом этого блока.

Модификатор — сущность, определяющая внешний вид, состояние и поведение блока или элемента. Например, внешний вид блока меню может изменяться в зависимости от применённого модификатора (мобильная версия, например, может быть реализована с большим шрифтом).

БЭМ используется не только в качестве нотации для именования CSS-классов, но и для организации блоков на файловой системе. Так, например, хранились блоки на начальных этапах разработки:



Некоторые шаблоны также делятся на две версии: версию для разработки и версию для сервера, поскольку серверу нужны свои управляющие структуры внутри кода шаблонов, чтобы он мог определить, куда и какие данные ему подставить. Естественно, что такие шаблоны не работают без сервера и зависят от данных, которые приходят от него.

## Инструменты

Поскольку разрабатываемое приложение многостранично, то необходим некоторые набор инструментов для облегчения разработки.

Во-первых, весь CSS-код, который получает клиент, изначально написан на SCSS, который потом транслируется препроцессором в то, что получает пользователь. SCSS позволяет повторно использовать отдельные CSS-правила без их копирования, а также помогает реализовать вложенную структуру правил, которой нет в CSS.

Во-вторых, после того, как в какой-то файл добавлены изменения, он автоматически проходит через систему сборки. Эта система написана на Gulp.js и реализует следующее:

* для HTML-файлов:
  + проверяет, есть ли в файле структуры типа @@include, и, если есть, то вставляет описанные ими файлы в текущий;
  + копирует полученные файлы в папку с готовыми шаблонами;
  + автоматически обновляет страницу с этим шаблоном, если она открыта в браузере.
* для SCSS-файлов:
  + запускает препроцессор SCSS, отдавая ему на вход изменённый файл;
  + копирует полученный файл в папку со скомпилированными CSS-файлами.
* для CSS-файлов:
  + анализирует CSS-правила на наличие в них тех, поддержка которых реализована частично, изменяет их структуру или добавляет для них вендорные префиксы, если это необходимо;
  + производит замену относительных путей для файлов;
  + выбирает media-правила и группирует их согласно описанным условиям;
  + минифицирует файл;
  + копирует полученный файл в папку с готовыми стилями;
  + автоматически обновляет страницу, на которой используется этот стиль, если она открыта в браузере.
* для изображений:
  + сжимает файл;
  + копирует полученный файл в папку с готовыми изображениями.
* для JS-файлов:
  + если это файлы в составе БЭМ-блоков, то:
    - соединяет их в один общий файл;
    - минифицирует;
    - копирует полученный файл в папку с готовыми скриптами;
  + если это какие-то другие JS-файлы, то:
    - минифицирует их;
    - копирует в папку с готовыми скриптами.
* копирует шрифты в папку с готовыми шрифтами;
* все прочие файлы копируются в специальную папку /files/.

На выходе мы получаем папку public, в которой лежат все файлы, которые необходимы для полноценной работы клиентской части с сервером и пользователем.

## Реализация

Клиентская часть состоит из следующих страниц:

* Авторизация — страница, на которой пользователь может войти, если он уже зарегистрирован, или зарегистрироваться.
* Список пользователей — страница со всеми пользователями, отсортированными по количеству подписчиков.
* Список стримов — список всех стримов, отсортированный по популярности.
* Лента (она же главная страниц) — последние обновления в стримах, на которые подписался пользователь.
* Добавление стрима — страница, на которой пользователь вписывает информацию о стриме и добавляет к нему теги и изображения. Она же (такая же внешне) открывается, если пользователь редактирует стрим.
* Добавление изображений — страница, на которой пользователь может добавить изображения, рассортировать их по стримам и добавить к ним теги.
* Настройки — страница с личной информацией пользователя и его настройками (например, отключение режима 18+).

Ссылки на эти страницы содержатся в верхнем меню сайта, которое общее на всех страницах.

# Заключение

В ходе данной курсовой работы был разработан веб-сервис для хранения, обмена и структурирования изображений. В ходе этой работы я закрепил свои знания, полученные в рамках курса «Конструирование программного обеспечения» и получил практический опыт работы с архитектурой клиент-сервер и системами контроля версий.