Домашнее задание 7

Дополнительные функции

Целью домашнего задания является исследование технологии Comet и механизма кэширования данных.

1. Real-time сообщения

Необходимо реализовать рассылку мгновенных сообщений о новых ответах всем пользователям, находящимся на странице определенного вопроса. Допустим есть два пользователя A и B. Оба находятся на странице одного вопроса, например **/question/33**. Пользователь A добавляет ответ на этот вопрос, пользователь B должен увидеть ответ без перезагрузки страницы.

Для этого необходимо:

- Настроить nginx-push-stream-module (https://www.nginx.com/resources/wiki/modules/push_stream/) в nginx (comet). Обратите внимание что этот модуль не относиться к стандартным. Nginx с этим модулем придется собирать самостоятельно (https://www.nginx.com/resources/wiki/modules/push_stream/#installation).
- На странице вопроса добавить JavaScript опрашивающий comet сервер.
- В форме добавления ответа добавить код, отправляющий сообщения в comet, например с помощью библиотеки requests.

2. Кэширование и фоновый запуск

Необходимо подготовить и вывести данные для правой колонки (лучшие пользователи, популярные тэги). Популярные тэги - это 10 тэгов с самым большим количеством вопросов за последние 3 месяца. Лучшие пользователи - это пользователи с самым большим количеством вопросов + ответов за последние 3 месяца.

Так как запросы на предполагаются тяжелыми, необходимо кэшировать данные на диске или memcached. Вьюшки не должны запускать эти запросы, а только брать данные из кэша. Для кэширования можно использовать встроенные механизмы Django.

Так как данные необходимы на каждой страницы, их придется загружать в каждой вьюшке, либо можно расширить шаблонизатор своими (inclusion) тэгами. Наполнять кэш данными необходимо с помощью Management команды (https://docs.djangoproject.com/en/1.10/howto/custom-management-commands/), запускаемой из Cron.

3. Полнотекстовый поиск

Необходимо реализовать поиск по заголовкам и содержимому вопросов. Пользователь вводит текст в поисковой строке, которая находиться в шапке. По введеному тексту СУБД должна находить совпадения, используя полнотекстовые индексы. Результаты поиска отображются пользователю в виде поисковых подсказок (выпадающий списокк под поисковой строкой).

Запрос должен отправляться автоматически по мере ввода пользователем частей текста. Необходимо удостовериться, что мы не перегружаем сервер лишними запросами, отправляя запрос на каждый новый введеный символ во время печати.

4. Полезные ссылки

- Документация по nginx-push-stream-module (https://www.nginx.com/resources/wiki/modules/push_stream/)
- Библиотека requests для python (http://docs.python-requests.org/en/latest/)
- Inclusion tags в Django (https://docs.djangoproject.com/en/1.10/howto/custom-template-tags/#inclusion-tags)
- Настройка кэширования в Django (https://docs.djangoproject.com/es/1.10/topics/cache/#filesystem-caching)
- Использование кэшей в Django (https://docs.djangoproject.com/es/1.10/topics/cache/#the-low-level-cache-api)
- Полнотекстовый поиск в MySQL (http://www.mysql.ru/docs/man/Fulltext_Search.html)
- И, наконец, cron (https://www.google.ru/webhp?sourceid=chrome-instant&ion=1&espv=2&ie=UTF-8&client=ubuntu#q=cron)