3. Запрос клиента записывает в базу данных информацию о нем (сервис, запрос, дата запроса). Так же данный запрос попадает в очередь запросов клиентов

4. Все запросы клиентов поочередно достаются из базы данных, после чего отправляются на сторонний ресурс и после прихода ответа от них – обратно записываются в базу данных к нужному запросу.

5. Клиенту в удобном визуализированном диаграммами виде показывается информация по его запросу, на основании которой он может провести SEO-аудит своего ресурса и выдвинуть дальнейшую стратегию продвижения.

В.А. Рубин (УО «ГГУ им Ф. Скорины», Гомель)
Науч. рук. **В.Д. Левчук**, канд. техн. наук, доцент

АСИНХРОННЫЕ ЗАДАНИЯ В DJANGO ДЛЯ ВЗАИМОДЕЙСТВИЯ С API СТОРОННИХ СЕРВИСОВ

В проекте, где необходимо вести постоянный обмен данными со сторонними сервисами посредством предоставляемого ими API, необходимо заранее приготовиться к серьезным нагрузкам на серверную часть.

В условиях высоких нагрузок недопустимо уже будет использовать синхронную модель взаимодействия, где ваш сервис произведет вызов на сторонний адрес, подождет от него ответ и потом произведет какие-либо операции по получении ответа. Ведь этого ответа можно не дождаться (недоступен ресурс), а возможно сервер просто «ляжет» под такими нагрузками.

Именно для таких нужд применяются асинхронные задания. Особенно красиво они реализованы в Python веб-фреймворке Django, а именно в одной из вспомогательных библиотек – Celery.

Celery – «distributed task queue». Это распределенная асинхронная очередь заданий, которая позволяет:

– выполнять задания асинхронно или синхронно, распределенно, одновременно,

– выполнять периодические и отложенные задания,

– ограничивать количество заданий в единицу времени (rate limit, для задания или глобально),

– мониторить выполнение заданий,

– присылать отчеты об ошибках на email,

– проверять, выполнилось ли задание (удобно для построения Ajax приложений, где клиент ждет факта завершения).

Для использования Celery в Django-проекте необходимо использовать сервис очередей. Ввиду особенностей проекта был выбран сервис RabbitMQ.

Такая организация проекта позволяет ему не «упасть» под высокой нагрузкой и не даст данным пользователя потеряться в случае неуспешного ответа со стороны сервиса.

Детали использования библиотеки Celery в производственном проекте обсуждаются в докладе.

А.Е. Руденко (УО «ГГУ им. Ф. Скорины», Гомель)
Науч. рук. **П.Л. Чечет,** канд. техн. наук

ФРЕЙМВОРК CORE DATA ДЛЯ ОПЕРАЦИОННОЙ СИСТЕМЫ IOS

Используя компьютеры для выполнения своих задач, люди рассчитывают, что внесенные ими изменения будет сохранены. Сохранение изменений играет важную роль в офисных программных пакетах, текстовых редакторах, играх, браузерах и т. д. Большинство программного обеспечения нуждается в возможности хранить введенные пользователем данные для последующего восстановления состояния работы, но конечно же есть и такое ПО, которое в этом не нуждается – калькуляторы, новостные ленты, будильники, виджеты о погоде. Понимание того, каким образом можно хранить данные на устройствах, является критически важным при разработке продвинутых приложений.

Core Data – это фреймворк для манипуляции с данными у графов объектов, предназначенный для операционных систем Mac OS X и iOS. Он был представлен для Mac OS X, начиная с версии 10.4 Tiger и для iOS, начиная с iPhone SDK 3.0. Фреймворк организовывает данные с помощью реляционной модели, позволяющей сериализовать аттрибуты сущностей в формат XML, бинарную форму или сохранять их в таблицы SQLite. Core Data является фреймворком для работы с данными, которая позволяет работать с сущностями и их связями (отношениями к другим объектами), атрибутами, в том виде, который напоминает работы с объектным графом в обычном объектно-ориентированном программировании. Т. е. данными можно манипулировать с помощью высокоуровневых объектов языка (например, Objective-C), которые представляют собой сущности предметной области и связи между ними.