Министерство образования и науки РФ
Федеральное государственное образовательное учреждение
высшего профессионального образования
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ
УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)
Кафедра комплексной информационной безопасности электронно-
вычислительных систем (КИБЭВС)

Место прохождения практики:
ТУСУР, каф. КИБЭВС, г.Томск

# ОТЧЕТ
# ПО РЕЗУЛЬТАТАМ
# производственной практики

Выполнила:
студентка гр. 722
_____ Мейта М.В.
«  »_____2015 г.

Руководитель практики от
предприятия:
к.т.н., доцент каф. КИБЭВС
_____ Романов А.С.
«  »_____2015 г.

Томск 2015

# Реферат

Отчёт по производственной практике содержит 35 страниц, 6 рисунков, 1 таблицу, 3 приложения.

PHYTON, ПРОГРАМНАЯ ПЛАТФОРМА DJANGO, TWITTER, API, GITHUB, ТВИТ, ХЭШТЕГ, SQLITE.

Целью технологической практики было начало разработки системы мониторинга настроения людей в социальных сетях для предотвращения массовых беспорядков и экстремизма, то есть такого программного комплекса, который позволит производить семантический анализ текстовых сообщений на предмет негативной эмоциональной окраски. Подобного рода система может быть использована в различных целях, как научно-исследовательских, так и коммерческих.

В качестве начального этапа создания данной системы должна была стать разработка программы для извлечения коротких текстовых сообщений («твитов») по определенному ключевому слову («хэштегу») в социальной сети «Twitter», последующего сохранения извлеченной информации в базу данных, а также веб-интерфейса для отображения полученных результатов.

В рамках данной практической работы моей задачей стало проектирование базы данных для извлечения необходимой информации, что в будущем понадобится для проведения семантического анализа. Была разработана структура базы данных, написаны соответствующие модели и шаблоны на основе программной платформы Django версии 1.8.0, произведены необходимые настройки и проведено исследование предметной области проблемы, а именно – семантического анализа тональности текста.

Планируется дальнейшая разработка программного комплекса для семантического анализа высказываний в социальных сетях, а также написание научно-исследовательской статьи в рамках X международной научно-практической конференции «Электронные средства и системы управления».

# СОДЕРЖАНИЕ

# Введение

В период 29 июня 2015 года по 12 июля 2015 года мною была пройдена технологическая практика на кафедре КИБЭВС ТУСУРа. В ходе прохождения практики была проведена следующая работа:

1) изучение предметной области проблемы, существующих разработок в области семантического анализа, инструментов и методов для решения поставленной задачи;

2) проектирование базы данных для хранения полученной выборки сообщений с сопутствующей ей информации об авторах, дате публикации и др.;

3) изучение программной платформы Django, ее установка и настройка;

4) разработка необходимых шаблонов для создания веб-интерфейса системы для отображения результатов работы программы, а именно – содержимого спроектированной базы данных.

# 1 Кафедра КИБЭВС

Местом прохождения практики была выбрана кафедра ТУСУРа – КИБЭВС (Кафедра комплексной информационной безопасности электронно-вычислительных систем).

Кафедра организована в ТУСУР в 1971 году как кафедра «Конструирования и производства электронно-вычислительной аппаратуры» (КиП ЭВА) вскоре переименованной в кафедру «Конструирования электронно-вычислительной аппаратуры» (КЭВА).

21 сентября 1999 г. в связи с открытием новой актуальной специальности 090105 – «Комплексное обеспечение информационной безопасности автоматизированных систем» кафедра КЭВА была переименована в кафедру «Комплексной информационной безопасности электронно-вычислительных систем» (КИБЭВС).

Заведующим кафедрой КИБЭВС на сегодняшний день является ректор ТУСУРа,

Александр Александрович Шелупанов, лауреат премии Правительства Российской Федерации, действительный член Международной Академии наук высшей школы РФ, действительный член Международной Академии информации, Почетный работник высшего профессионального образования РФ, заместитель Председателя Сибирского регионального отделения учебно-методического объединения вузов России по образованию в области информационной безопасности, профессор, доктор технических наук.

С 2008 г. кафедра КИБЭВС входит в состав Института «Системной интеграции и безопасности».

На базе кафедры КИБЭВС ТУСУР в 2002 году организовано «Сибирское региональное отделение учебно-методического объединения Вузов России по образованию в области информационной безопасности [1].

# 2 Анализ современных исследований в области семантического анализа тональности высказываний

Проблема определения тональности (эмоциональной окраски) текста на сегодняшний день вызывает широкий интерес в различных областях, будь то маркетинг, предсказание трендов на рынке ценных бумаг, научные исследования или какие-либо иные цели, подразумевающие интеллектуальный анализ текстов. Богатой почвой для исследований могут стать социальные сети, набравшие огромную популярность среди людей по всему миру и дающие возможность практически каждому жителю на планете высказывать некие мнения по тем или иным вопросам. Прекрасной базой для семантического анализа является социальная сеть «Twitter», где пользователи публикуют короткие текстовые сообщения (так называемые «твиты») длиной до 140 символов, зачастую являющиеся эмоционально окрашенными высказываниями, полезными в различных социологических и маркетинговых исследованиях. За счет микроблогинга, а также наличия доступных для разработчиков API-функций, предоставляемых самой компанией Twitter Inc, выбор платформы для начала проектирования системы был сделан в пользу именно данной социальной сети.

Существует множество разработок в области семантического анализа и создания нейронных сетей, способных решать задачи компьютерной лингвистики. Среди них – разработанная учеными из Стэнфорда нейросеть, способная, по заявлению разработчиков, определять тональность англоязычного текста с точностью 85% [2]. Также есть различные отечественные разработки – как коммерческие, так и доступные для исследования и научной работы. Однако по-прежнему данная тема остается актуальной и интерес к ней активно возрастает.

Классификация высказываний по эмоциональной окраске может быть проведена различными способами. Условно можно разделить все высказывания на три подмножества: позитивно, негативно или нейтрально

окрашенные. Количество классов, на которые делят тональность, обычно задается из спецификации системы, то есть зависит от целей, преследуемых при ее создании. Далее встает проблема обучения автоматизированной системы семантическому анализу тональности произвольного высказывания из конечного множества сообщений.

«Sentiment analysis (по-русски, анализ тональности) — это область компьютерной лингвистики, которая занимается изучением мнений и эмоций в текстовых документах. Целью анализа тональности является нахождение мнений в тексте и определение их свойств. В зависимости от поставленной задачи нас могут интересовать разные свойства, например, автор — кому принадлежит это мнение, тема — о чем говорится во мнении, тональность — позиция автора относительно упомянутой темы (обычно «положительная» или «отрицательная»)» [3].

Можно выделить несколько подходов в анализе тональности текстов:

1) основанные на правилах – набор правил, позволяющих сделать заключение о тональности текста; основной недостаток такого подхода – для хорошей работы системы необходимо составить большое количество правил;

2) основанные на словарях – используют так называемые тональные словари для анализа текста; в простом виде тональный словарь представляет собой список слов со значением тональности для каждого слова;

3) машинное обучение с учителем – наиболее распространенный метод, суть которого состоит в том, чтобы обучить машинный классификатор на коллекции заранее размеченных текстов, а затем использовать полученную модель для анализа новых документов;

4) машинное обучение без учителя – наименее точный метод анализа, в ходе которого система «обучается» самостоятельно, следовательно, существует большая вероятность ошибочной классификации [3].

В любом случае неотъемлемой частью создания подобного рода системы является наличие данных для обучения и заранее определенная тема исследований для проведения единовременного анализа. Выбор алгоритмов

классификации, набора исследуемых признаков, методов обработки полученной информации и форма входных данных определяются самими разработчиками и заинтересованными лицами.

Среди отечественных разработок, свободно доступных для исследования и применения в собственных проектах, можно выделить проект SentiRuEval-2015, предназначенный для анализа отзывов о ресторанах и автомобилях [4]. SentiRuEval представляет собой набор скриптов на языке программирования Python и подробное теоретическое описание возможностей системы. Также разработчики предоставляют пример входных данных для анализа и инструкцию по запуску SentiRuEval.

В аннотации к научной статье [5] цели создания SentiRuEval определены следующим образом: «Участникам были предложены два задания. Первым заданим был аспектно-ориентированный анализ отзывов о ресторанах и автомобилях; основная цель этого задания состояла в поиске слов и выражений, обозначающих важные характеристики сущности (аспектные термины), и классификации их по тональности и обобщенным категориям. Второе задание заключалось в анализе влияния твитов на репутацию заданных компаний. Такие твиты могут либо выражать мнение пользователя о компании, ее продукции или услугах, или содержать негативные или позитивные факты, которые стали известны об этой компании». Приведенное описание представляет существенный интерес для разработки системы мониторинга настроения людей в социальных сетях с целью предотвращения массовых беспорядков и экстремизма, и будет изучена подробнее в дальнейшем.

# 3 Разработка системы мониторинга настроения людей в социальных сетях с целью предотвращения массовых беспорядков и экстремизма

В качестве первого шага в разработке данной системы была поставлена задача создать программу для извлечения коротких текстовых сообщений («твитов»), публикуемых в социальной сети «Twitter» по ключевому слову («хэштегу») с последующим сохранением в базу данных как самих сообщений, так и сопутствующей им информации (об авторе, дате публикации, локации пользователя и др.). Также разработать веб-интерфейс для отображения полученных данных и продолжить исследование данной предметной области для написания научно-исследовательской статьи в рамках X Международной научно-практической конференции «Электронные средства и системы управления» [6] и дальнейшей разработки системы, включающей в себя разработку системы анализа тональности сообщений.

Архитектура системы в целом для дальнейших разработок примет вид, представленный на рисунке 3.1. С клиентской стороны пользователь переходит по определенной URL, в результате чего формируется GET-запрос на сервер хоста. Все возможные ссылки данного сайта в Django хранятся в файле urls.py в виде регулярных выражений. Далее запрашиваемая URL-ссылка обрабатывается соответствующей view-функцией (views.py), находится html-шаблон, который, в свою очередь, наследует ряд других html, css и js-шаблонов, и в результате клиент видит запрашиваемую страницу.

Для работы с базой данных в Django используются модели, создаваемые разработчиком и Python DB API.

Среди функций представления также будут функции для семантического анализа и поиска твитов с сохранением в базу данных. Эти же функции будут возвращать клиенту в качестве HttpResponse-ответа полученные при анализе данные.

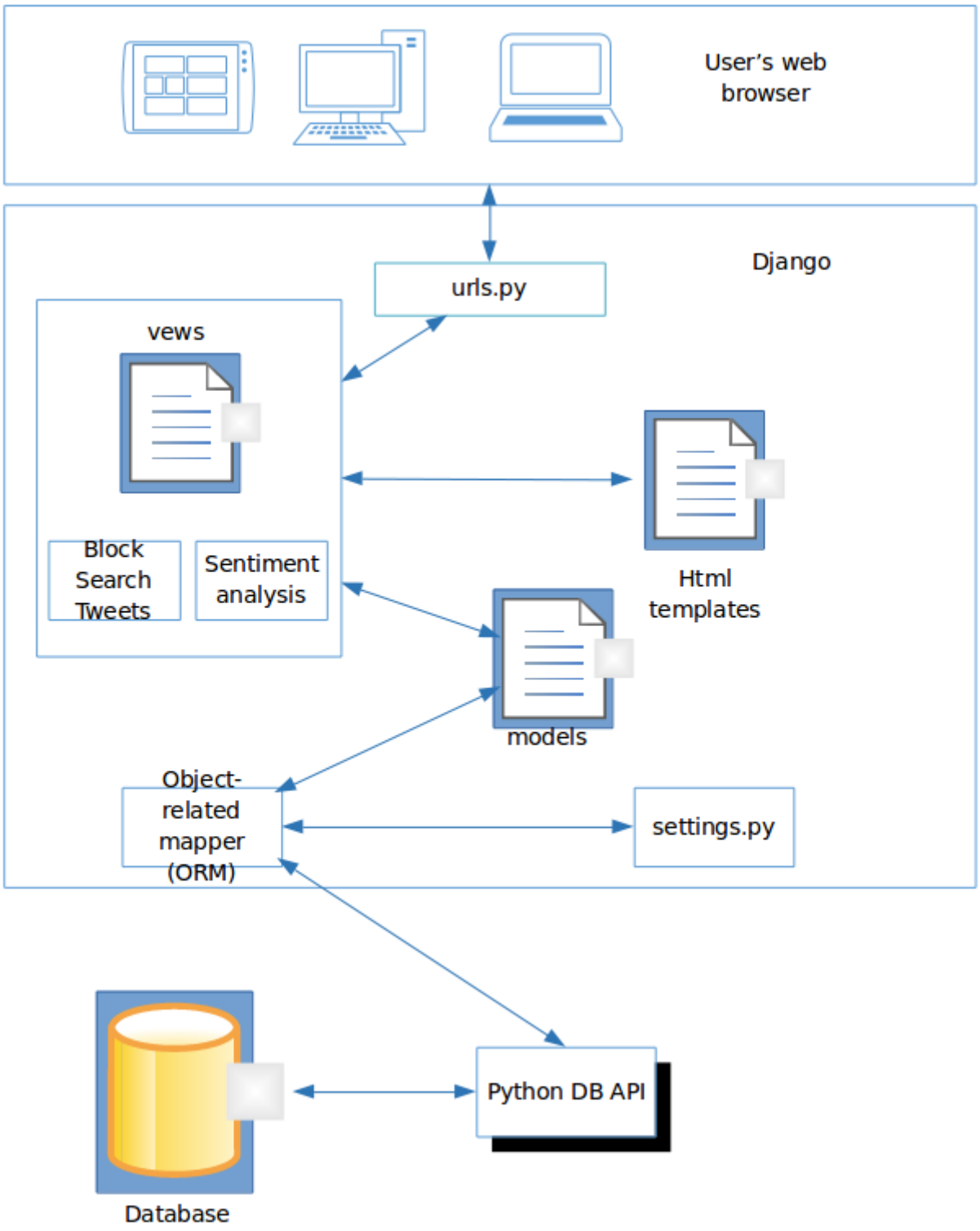


Рисунок 3.1 – Архитектура системы мониторинга настроения людей в социальных сетях

## 3.1 Используемые программные средства, обоснование выбора и их описание

В качестве инструментов разработки были выбраны:

1) интерпретируемый высокоуровневый язык программирования Python версии 2.7.9 [7];

2) встраиваемая реляционная база данных SQLite 3;

3) программная платформа Django версии 1.8.0;

4) система контроля версий Git и веб-сервис для создания удаленного репозитория GitHub;

5) кросплатформенный текстовый редактор Sublime Text 2 для написания кода;

6) ОС Linux (Ubuntu 14.10).

### 3.1.1 Встраиваемая реляционная база данных SQLite

В качестве базы данных для хранения данных, извлекаемых приложением, была выбрана SQLite 3, так как она является компактной, встраиваемой, легко переносимой и производительной. Встраиваемая означает, что SQLite не использует парадигму клиент-сервер, движок SQLite не является отдельно работающим процессом, с которым взаимодействует программа, а предоставляет библиотеку, с которой программа компонуется, и движок становится составной частью программы. SQLite считывает и хранит данные в виде обыкновенных файлов. В этом случае база данных с множеством различных таблиц, индексов, триггеров и представлений содержится в одном единственном файле на жестком диске.

Формат такой базы данных является кросс-платформенным, позволяет свободно переносить (копировать) базу данных с 32-битных на 64-битные системы и наоборот, а также между big-endian и little-endian архитектурами.

SQLite – компактная библиотека. При включении всех возможных расширений ее размер может составлять менее 500 килобайт в памяти компьютера, в зависимости от используемой платформы и настроек оптимизации компилятора.

SQLite также может запускаться на устройствах с маленьким пространством стэка (4 Кб) и кучей (100 Кб), что делает SQLite популярным движком для различных гаджетов вроде смартфонов, MP3-плееров и т.д.

SQLite является бестиповой базой данных. Точнее, есть только два типа – целочисленный («integer») и текстовый («text»). Причём «integer» используется преимущественно для первичного ключа таблицы, а для остальных данных – «text». Длина строки, записываемой в текстовое поле, может быть любой. Поскольку движок базы и интерфейс к ней реализованы как единое целое, огромный преимуществом SQLite является высокая производительность [8].

### 3.1.2 Программная платформа Django

Django — свободный фреймворк (программная платформа) для веб-приложений на языке Python, использующий шаблон проектирования MVC. Проект поддерживается организацией Django Software Foundation.

Сайт на Django строится из одного или нескольких приложений, которые рекомендуется делать отчуждаемыми и подключаемыми. Это одно из существенных архитектурных отличий этого фреймворка от некоторых других (например, Ruby on Rails). Один из основных принципов фреймворка — DRY (англ. Don't repeat yourself)

Также, в отличие от других фреймворков, обработчики URL в Django конфигурируются явно при помощи регулярных выражений, а не выводятся автоматически из структуры моделей контроллеров.

Для работы с базой данных Django использует собственный ORM, в котором модель данных описывается классами Python, и по ней генерируется схема базы данных.

Архитектура Django похожа на «Модель-Представление-Контроллер» (MVC). Контроллер классической модели MVC примерно соответствует уровню, который в Django называется Представление (англ. View), а презентационная логика Представления реализуется в Django уровнем Шаблонов (англ. Template). Из-за этого уровневую архитектуру Django часто называют «Модель-Шаблон-Представление» (MTV).Первоначальная разработка Django, как средства для работы новостных ресурсов, достаточно сильно отразилась на его архитектуре: он предоставляет ряд средств, которые помогают в быстрой разработке веб-сайтов информационного характера. Так, например, разработчику не требуется создавать контроллеры и страницы для административной части сайта, в Django есть встроенное приложение для управления содержимым, которое можно включить в любой сайт, сделанный на Django, и которое может управлять сразу несколькими сайтами на одном сервере. Административное приложение позволяет создавать, изменять и

удалять любые объекты наполнения сайта, протоколируя все совершённые действия, и предоставляет интерфейс для управления пользователями и группами (с пообъектным назначением прав) [9].

Преимущества использования Django:

– использование Python в качестве языка программирования, а следовательно, все преимущества данного языка;

– качественная документация;

– встроенный ORM (Object-relational mapper) – технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных»;

– автоматически генерируемый интерфейс администратора;

– поддержка MVT (Model-Template-View) – паттерн проектирования, предполагающий использование моделей, шаблонов и представлений (view-функций);

– наличие встроенного обработчика ошибок;

– удобство настройки;

– широкое коммьюнити разработчиков;

– высокая скорость работы.

### 3.1.3 Система контроля версий Git и веб-сервис для создания удаленного репозитория GitHub

Git — распределённая система управления версиями. Проект был создан Линусом Торвальдсом для управления разработкой ядра Linux, первая версия выпущена 7 апреля 2005 года. Программа является свободной и выпущена под лицензией GNU GPL версии 2 [10].

При работе над одним проектом как команде разработчиков, так и отдельного программиста, необходим инструмент для совместного написания, бэкапирования и тестирования программного обеспечения. Используя Git, мы имеем:

– возможность удаленной работы с исходными кодами;

– возможность создавать свои ветки, не мешая при этом другим разработчикам;

– возможность бэкапирования проекта;

– доступ к последним изменениям в коде;

– возможность откатиться к любой стабильной стадии проекта.

Для создания удаленного репозитория использовался веб-сервис GitHub.

GitHub – веб-сервис для хостинга IT-проектов и их совместной разработки. Основан на системе контроля версий Git и разработан на Ruby on Rails и Erlang компанией GitHub, Inc (ранее Logical Awesome). Сервис абсолютно бесплатен для проектов с открытым исходным кодом

Создатели сайта называют GitHub «социальной сетью для разработчиков». Кроме размещения кода, участники могут общаться, комментировать правки друг друга, а также следить за новостями знакомых. С помощью широких возможностей Git программисты могут объединять свои репозитории — GitHub предлагает удобный интерфейс для этого и может отображать вклад каждого участника в виде дерева [11].

# 4 Создание и настройка шаблона проета в Django

Для начала необходимо было:

1) создать директорию для хранения проекта
   $ mkdir dir_name
2) создать виртуальную изолированную оболочку и запустить ее:
   $ virtualenv –python=python 2.7.9 djenv
   $ source djenv/bin/activate
3) установить в нее Django
   $ pip install django==1.8
4) создать и настроить проект:
   $ dlango-admin startproject mysite .
   $ python manage.py startapp web_app

После создания основного шаблона проекта были прописаны модели, шаблоны и представления, необходимые для осуществления поставленной задачи. Рассмотрим подробнее принципы и особенности работы Django.

При начальном запуске сервера соответствующий скрипт ищет в текущем каталоге (в том же, где находится manage.py) файл settings.py, где указаны основные настройки проекта. Оттуда он считывает основные параметры, такие как DATABASES (параметры соединения с БД) и TEMPLATES (директории с html-шаблонами и стилями CSS):

```
DATABASES = {
   'default': {
      'ENGINE': 'django.db.backends.sqlite3',
      'NAME': os.path.join(BASE_DIR, 'db/test.db'),
   }
}
TEMPLATES = [{
      'BACKEND': 'django.template.backends.django.DjangoTemplates',
      'DIRS': [os.path.join(BASE_DIR, 'static', 'templates')],
      'APP_DIRS': True,
```

```
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },]
```

Самым главным параметром является ROOT_URLCONF, который указывает Django, какой модуль Python следует использовать в качестве привязки для данного сайта. Когда приходит запрос на определенный URL, Django загружает файл привязок, указанный параметром ROOT_URLCONF:

```
ROOT_URLCONF = 'web_interface.urls'
```

Затем Django проверяет каждый шаблон этого файла по порядку, сравнивая запрошенный URL с шаблонами, пока не найдет подходящий.

Шаблоны URL, представленные в виде регулярных выражений:

```
urlpatterns = [
    # Examples:
    url(r'^$', 'web_app.views.home', name='home'),
    url(r'^search', 'web_app.views.search', name='search'),
    # url(r'^blog/', include('blog.urls')),
    url(r'^admin/', include(admin.site.urls)),
]
```

Если совпадение найдено, Django вызывает функцию представления, ассоциированную с шаблоном, передавая ей HttpRequest в качестве первого аргумента. Функция представления должна возвращать HttpResponse. В данном случае она обращается к шаблону, находящемуся в директории, указанной в TEMPLATES. Функция загружает базовую страницу, а затем – страницу-наследника.

Удобство использования шаблонов заключается, разумеется, в том, что разработчик не тратит время на повторное написание или копирование кода,

а значит, выполняется основной постулат Django – DRY (Don`t Repeat Yourself – «не повторяй себя»), что позволяет сэкономить время на разработку, избежать ошибок и иметь удобную и логичную структуру всего проекта в целом.

Базовый шаблон html-страницы, который, в свою очередь, подгружает шаблоны-наследники при помощи блочных тегов, содержит следующий программный код:

```
{% load staticfiles %}
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <link rel="stylesheet" href="{% static "css/base_css.css" %}">
  </head>
    <body>
      <form action="/search/" method="post">
        <input type="text" id="fieldsearch"size="30">
        {% csrf_token %}
        <button id="search" >Поиск</button>
      </form>
      <form action="/" method="post">
        {% csrf_token %}
        <button id="refreshtable">Обновить таблицу</button>
      </form>
      <table>
        <tr>
          <th>id</th>
          <th>text</th>
          <th>hashtags</th>
          <th>created at</th>
        </tr>
        {% for item in latest_question_list %}
        <tr>
          <td>{{ item.tweet_id}}</td>
```

```
            <td>{{ item.tweet_text}}</td>
            <td>{{ item.hashtags}}</td>
            <td>{{ item.created_at}}</td>
        </tr>
        {% endfor %}
    </table>
</body>
```

Блок-схема описанного процесса работы Django представлена на рисунке 4.1.

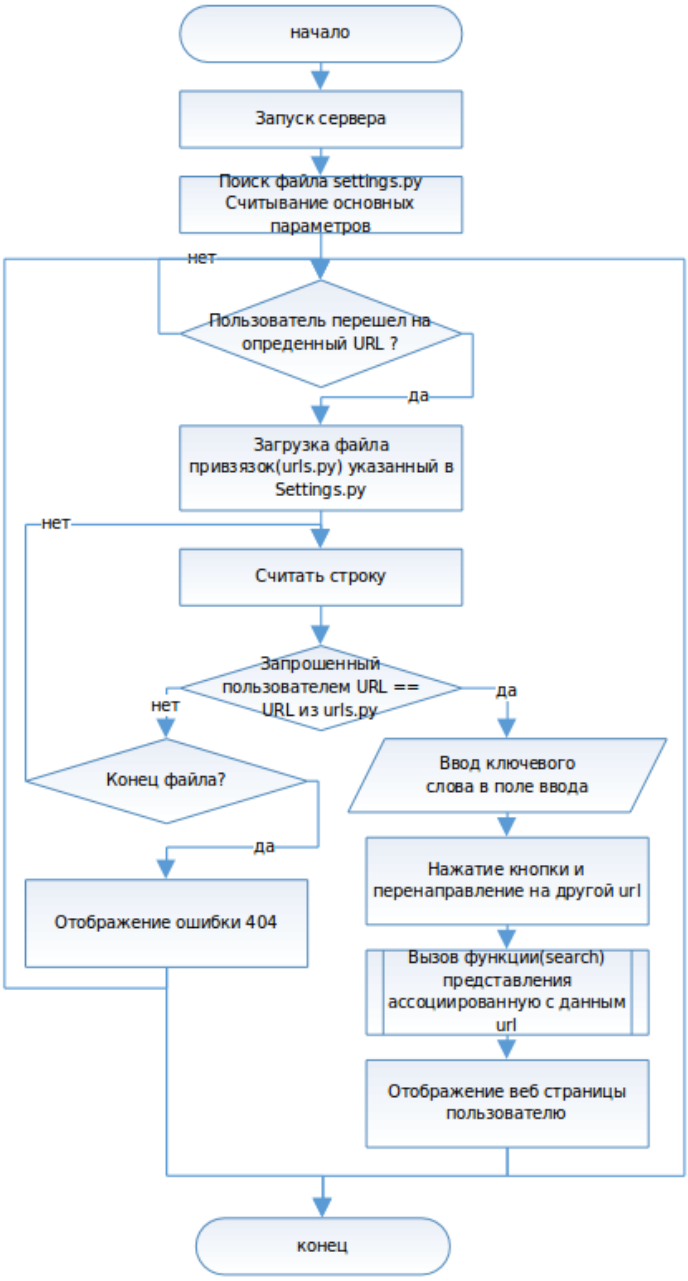


Рисунок 4.1 – Блок-схема процесса работы Django

# 5 Проектирование базы данных. Создание моделей в Django

Модели отображают информацию об обрабатываемых данных и содержат поля и поведение этих данных. Обычно одна модель представляет одну таблицу в базе данных и описывается классом python. С помощью данного класса существует возможность создавать, получать, обновлять и удалять записи в таблице базы данных, используя простой код на языке Python вместо использования повторяющихся SQL-команд.

Сравнительно небольшой код модели дает Django достаточно много информации. На основании которой Django сможет сделать следующее:

1) создать структуру базы данных (операторы CREATE TABLE);

2) создать набор API функции для взаимодействия между БД и объектами.

Имена таблиц генерируются автоматически и получаются путем соединения имени приложения и имени модели.

Особенности моделей в Django:

1) каждая модель — это класс, унаследованный от модели django.db.models.Model;

2) каждый атрибут модели представляет собой поле в базе данных;

3) Django предоставляет автоматически созданное API для доступа к данным.

После определения моделей необходимо указать Django, что необходимо их использовать. Сделать это можно, отредактировав файл настроек и изменив INSTALLED_APPS, добавив пакет, который содержит models.py.

Самая важная часть модели – и единственная обязательная – это список полей таблицы базы данных, которые она представляет. Поля определены атрибутами класса.

Каждое поле в модели должно быть экземпляром соответствующего Field-класса. Django использует классы полей для определения следующей информации:

– типа колонки в базе данных (например: INTEGER, VARCHAR);

– widget для использования в интерфейсе администратора, если планируется его использование;

– минимальные правила проверки данных, используемые в интерфейсе администратора и для автоматического создания формы.

Первичный автоинкрементный ключ по умолчанию задается полем модели, например, такого вида:

id = models.AutoField(primary_key=True).

Поскольку скрипт, который соединяется с сервером с помощью Twitter-API, сохраняет в базу не только сами сообщения, но также информацию об авторе, местоположении, времени опубликования и др., таблица для хранения этих данных примет следующий вид (таблица 5.1). Диаграмма таблицы «Твиты» в методологии IDEF1X представлена на рисунке 5.1.

Таблица 5.1 – Таблица признаков для Tweets

| Наименование признака | Описание признака | Диапазон допустимых значений |
|---|---|---|
| Tweet_id | Уникальный номер твита | Большое целое, первичный ключ |
| Tweet_text | Текст твита | 140 символов (максимальная длина твита) |
| Hashtags | Хештеги, используемые в твите | Нет ограничений |
| Created_at | Время и дата опубликования | Нет ограничений |

| | | |
|---|---|---|
| User_name | Имя пользователя | Нет ограничений |
| Lang | Язык сообщения | Нет ограничений |
| Time_zone | Часовой пояс | Нет ограничений |
| Location | Местоположение | Нет ограничений |

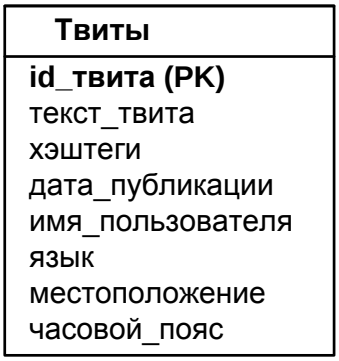


Рисунок 5.1 – Диаграмма таблицы «Твиты» в методологии IDEF1X

Django использует миграции для переноса изменений в моделях (добавление поля, удаление модели и т.д.) на структуру базы данных. Миграции создавались в основном для автоматической работы, но необходимо знать, когда их создавать, запускать и как решать различные проблемы. Модели создаются в файле models.py (приложение А) и «мигрируют» в базу данных с помощью специального скрипта manage.py.

До версии 1.7, Django позволял только добавлять новые модели в базу данных; не было возможности изменять или удалять существующие модели, используя команду syncdb (предок команды migrate).

Сторонние инструменты позволяли создавать и выполнять миграции. Со временем было решено перенести этот функционал в Django.

Django предоставляет две команды для работы с миграциями и структурой базы данных:

– migrate, которая отвечает за применение миграций, за откат миграций и за вывод статуса миграций;

– makemigrations, которая отвечает за создание новых миграций на основе изменений в моделях;

– sqlmigrate, которая выводит SQL запросы для миграции.

Стоит отметить, что миграции создаются и работают в контексте отдельного приложения. В частности, можно создать приложение, которое не использует миграции - такие приложения имитируют старое поведение и просто создают новые модели.

Следует рассматривать миграции, как систему контроля версий для базы данных. Makemigrations отвечает за сохранение состояния моделей в файле миграции - аналог коммита - а migrate отвечает за их применение к базе данных.

Файлы с миграциями находятся в каталоге “migrations” приложения. Они являются частью приложения и должны распространятся вместе с остальным кодом приложения. Они должны создаваться при разработке и потом применятся на машинах коллег, тестовом и “боевом” серверах.

Миграции поддерживаются всеми бэкендами, которые предоставляет Django, как и сторонними, если они реализуют API внесения изменений в структуру базы данных (через класс SchemaEditor).

Тем не менее, некоторые базы данных поддерживают больше возможностей, чем другие, в случаях когда речь идёт о миграциях схемы.

SQLite очень плохо поддерживает изменения в структуре базы данных, но Django пытается эмулировать их следующим образом:

– создание новой таблицы для новой структуры;

– копирование данных в новую таблицу;

– удаление старой таблицы;

– переименование новой таблицы.

Этот процесс как правило хорошо работает, но может быть медленным. Не рекомендуется использовать и мигрировать SQLite на «боевом» сервере, если вы не очень осведомлены о рисках и его ограничениях. Django поддерживает SQLite, чтобы позволить разработчикам использовать SQLite для разработки простых проектов.

Содержание полученной таблицы в базе данных можно увидеть на рисунке 5.2.

| rowid | tweet_id | tweet_text | hashtags | created_at | user_name | lang | time_zone | location |
|---|---|---|---|---|---|---|---|---|
| 1 | 6394122... | 3 сентября. П... | 3сентября,с... | 2015-09-03 12:... | Irina Antip... | ru | Bangkok | Tomsk |
| 4 | 6387773... | Началась нов... | тусур | 2015-09-01 18:... | Вероника... | ru | None | Томск |
| 5 | 6386485... | Отличный ден... | ТУСУР,Том... | 2015-09-01 09:... | Irina Antip... | ru | Bangkok | Tomsk |
| 6 | 6386364... | Теперь я с ва... | РКФ,ТУСУР | 2015-09-01 08:... | Marina Se... | ru | Bangkok | |
| 7 | 6385695... | Все на парах, ... | каникулы,с... | 2015-09-01 04:... | Ксения Гл... | ru | Bangkok | Северск |
| 10 | 6372846... | Завтра уже ед... | ТУСУР,LIVE | 2015-08-28 15:... | Максим ... | ru | Abu Dhabi | Прокопьевск-Томск |
| 11 | 6369289... | Откапал свою... | такт,тусур,т... | 2015-08-27 15:... | Dmitry Le... | ru | Moscow | Tyumen \| Salekhard \| ... |
| 12 | 6391195... | #профсоюз #т... | профсоюз,т... | 2015-09-02 16:... | не рич бич | ru | Bangkok | |

Рисунок 5.2 – Содержание таблицы базы данных с полученными данными

# 6 Результаты работы

В ходе совместной работы с другим разработчиком над данным проектом был создан шаблон проекта на программной платформе Django, написано веб-приложение для поиска твитов по ключевому слову с последующим сохранением в базу данных, продумана структура базы данных и необходимые модели, произведено исследование предметной области вопроса, подготовлена модель проекта для дальнейшей разработки, создан веб-интерфейс для отображения полученных результатов.

Результат работы веб-приложения представлен на рисунках 6.1 и 6.2.

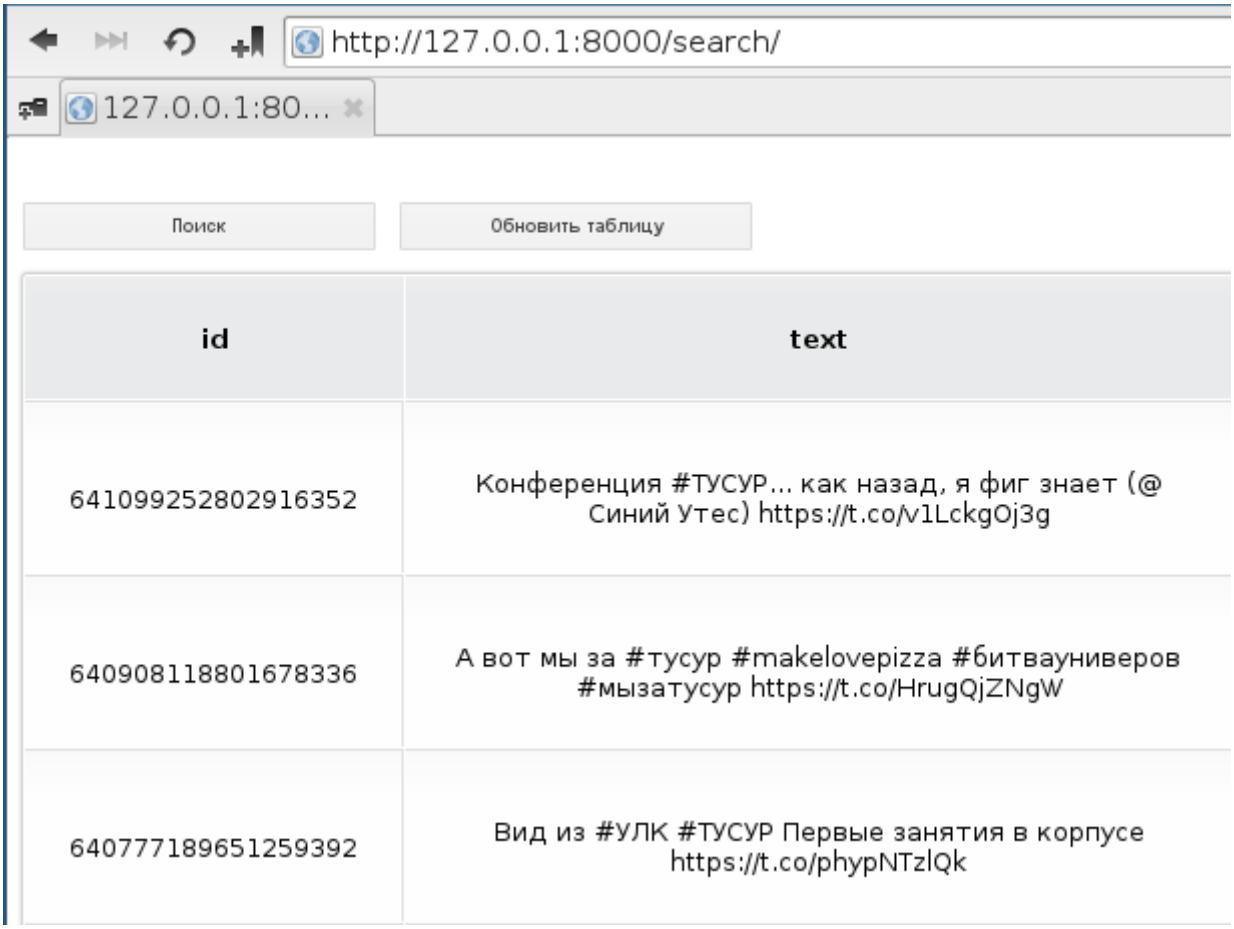


Рисунок 6.1 – Результат работы веб-приложения для поиска и сохранения твитов

| hashtags | created at | username | location |
|---|---|---|---|
| ТУСУР | 2015-09-08 04:02:27 | Павел Тамронский | Tomsk (Томск) Россия |
| тусур,makelovepizza,битвауниверов,мызатусур | 2015-09-07 15:22:57 | Jack Black | Томск |
| УЛК,ТУСУР | 2015-09-07 06:42:41 | Irina Serebrennikova | Tomsk |

Рисунок 6.2 – Результат работы веб-приложения для поиска и сохранения твитов (продолжение)

# Заключение

В результате прохождения производственной практики было проделано следующее:

1) изучена предметная область проблемы, проведен обзор существующих разработок в области семантического анализа, инструментов и методов для решения поставленной задачи;

2) спроектирована база данных для хранения полученной выборки сообщений с сопутствующей ей информации об авторах, дате публикации и др.;

3) изучена программная платформы Django, осуществлена ее установка и настройка;

4) разработаны необходимые шаблоны для создания веб-интерфейса системы для отображения результатов работы программы, а именно – содержимого спроектированной базы данных.

Текущую версию разрабатываемого проекта можно посмотреть и загрузить с репозитория на GitHub [12].

# Список использованных источников

1 Официальный сайт КИБЭВС [Электронный ресурс]. – Режим доступа: http://kibevs.tusur.ru/pages/kafedra/index (дата обращения: 9.07.2015)

2 The Stanford Natural Language Processing Group [Электронный ресурс]. – Режим доступа: http://nlp.stanford.edu/software/corenlp.shtml (дата обращения: 30.06.2015)

3 Обучаем компьютер чувствам (sentiment analysis по-русски) [Электронный ресурс]. – Режим доступа: http://habrahabr.ru/post/149605 (дата обращения: 9.07.2015)

4 SentiRuEval-2015 (Google Диск) [Электронный ресурс]. – Режим доступа: https://drive.google.com/folderview?id=0B7y8Oyhu03y_fjNIeEo3UFZObTVDQXBrSkNxOVlPaVAxNTJPR1Rpd2U1WEktUVNkcjd3Wms&usp=drive_web (дата обращения: 9.07.2015)

5 Лукашевич Н. В. SentiRuEval: тестирование систем анализа тональности текстов на русском языке по отношению к заданному объекту / Н. В. Лукашевич, П. Д. Блинов, Е. В. Котельников, Ю. В. Рубцова, В. В. Иванов, Е. Тутубалина // Компьютерная лингвистика. – 2015. – 13 с.

6 X международная научно-практическая конференция «Электронные средства и системы управления» (ТУСУР) [Электронный ресурс]. – Режим доступа: http://www.tusur.ru/ru/scienceevents/conferences (дата обращения: 30.06.2015)

7 Python Software Foundation. Official website [Электронный ресурс]. – Режим доступа: https://www.python.org (дата обращения: 1.07.2015)

8 SQLite Home Page [Электронный ресурс]. – Режим доступа: https://www.sqlite.org/ (дата обращения: 1.07.2015)

9 Django official website [Электронный ресурс]. – Режим доступа: https://www.djangoproject.com (дата обращения: 1.07.2015)

10 Git official website [Электронный ресурс]. – Режим доступа: https://git-scm.com (дата обращения: 29.06.2015)

11 GitHub [Электронный ресурс]. – Режим доступа: https://github.com (дата обращения: 29.06.2015)

12 Наш проект на GitHub [Электронный ресурс]. – Режим доступа: https://github.com/MarinaMeyta/social_network_analisys analysis (дата обращения: 2.07.2015)

13Образовательный стандарт ВУЗа. – Стандарт для студентов ВУЗа. – Томск, 2013. – 57 с.

14 Методические указания по проведению производственной практики. – Учебное пособие для студентов ВУЗа. – Томск, 2015. – 32 с.

# Приложение А

## Содержание файла models.py

```
from django.db import models

# Create your models here.
class Tweets(models.Model):
	tweet_id = models.BigIntegerField(primary_key = True)
	tweet_text = models.CharField(max_length = 140)
	hashtags = models.TextField()
	created_at = models.TextField()
	user_name = models.TextField()
	lang = models.TextField()
	time_zone = models.TextField()
	location = models.TextField()
```

# Приложение Б

## Содержание файла settings.py

```
# coding: utf-8
"""
Django settings for web_interface project.
Generated by 'django-admin startproject' using Django 1.8.
For more information on this file, see
https://docs.djangoproject.com/en/1.8/topics/settings/
For the full list of settings and their values, see
https://docs.djangoproject.com/en/1.8/ref/settings/
"""
# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
import os
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))

# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/1.8/howto/deployment/checklist/
# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'ska*ydpea__1bf9j*4f(i^-f!vkzhw7mvv1o-!37qf^65z!tyh'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []

# Application definition
INSTALLED_APPS = (
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'web_app',

)

MIDDLEWARE_CLASSES = (
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.auth.middleware.SessionAuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
    'django.middleware.security.SecurityMiddleware',
)

ROOT_URLCONF = 'web_interface.urls'
```

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR, 'static', 'templates')],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'web_interface.wsgi.application'

# Database
# https://docs.djangoproject.com/en/1.8/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db/test.db'),
    }
}

# Internationalization
# https://docs.djangoproject.com/en/1.8/topics/i18n/

LANGUAGE_CODE = 'ru-ru'
TIME_ZONE = 'UTC'
USE_I18N = True
USE_L10N = True
USE_TZ = True

# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/1.8/howto/static-files/

STATIC_URL = '/static/'
STATIC_ROOT = os.path.join(BASE_DIR, 'static')
STATICFILES_DIRS = (
    os.path.join(BASE_DIR, "static", "css"),
)
```

# Приложение В

## Содержание файла base_css.css

```
#search {
  width: 250px;
  height: 22px;
  display: inline-block;
  font-family: arial,sans-serif;
  font-size: 11px;
  font-weight: bold;
  color: rgb(68,68,68);
  text-decoration: none;
  user-select: none;
  padding: .2em 1.2em;
  outline: none;
  border: 1px solid rgba(0,0,0,.1);
  border-radius: 2px;
  background: rgb(245,245,245) linear-gradient(#f4f4f4, #f1f1f1);
  transition: all .218s ease 0s;
}

#search:hover {
  color: rgb(24,24,24);
  border: 1px solid rgb(198,198,198);
  background: #f7f7f7 linear-gradient(#f7f7f7, #f1f1f1);
  box-shadow: 0 1px 2px rgba(0,0,0,.1);
}

#search:active {
  color: rgb(51,51,51);
  border: 1px solid rgb(204,204,204);
  background: rgb(238,238,238) linear-gradient(rgb(238,238,238), rgb(224,224,224));
  box-shadow: 0 1px 2px rgba(0,0,0,.1) inset;
}

#refreshtable {
  width: 250px;
  height: 22px;
  display: inline-block;
  font-family: arial,sans-serif;
  font-size: 11px;
  font-weight: bold;
  color: rgb(68,68,68);
  text-decoration: none;
  user-select: none;
  padding: .2em 1.2em;
  outline: none;
  border: 1px solid rgba(0,0,0,.1);
  border-radius: 2px;
  background: rgb(245,245,245) linear-gradient(#f4f4f4, #f1f1f1);
```

```
  transition: all .218s ease 0s;
}

#refreshtable:hover {
  color: rgb(24,24,24);
  border: 1px solid rgb(198,198,198);
  background: #f7f7f7 linear-gradient(#f7f7f7, #f1f1f1);
  box-shadow: 0 1px 2px rgba(0,0,0,.1);
}

#refreshtable:active {
  color: rgb(51,51,51);
  border: 1px solid rgb(204,204,204);
  background: rgb(238,238,238) linear-gradient(rgb(238,238,238), rgb(224,224,224));
  box-shadow: 0 1px 2px rgba(0,0,0,.1) inset;
}

table {
    position:relative;
    top: -6%;
    table-layout: auto;
    overflow:hidden;
    border:1px solid #d3d3d3;
    background:#fefefe;
    width:100%;
    margin:5% auto 0;
    -moz-border-radius:5px; /* FF1+ */
    -webkit-border-radius:5px; /* Saf3-4 */
    border-radius:5px;
    -moz-box-shadow: 0 0 4px rgba(0, 0, 0, 0.2);
    -webkit-box-shadow: 0 0 4px rgba(0, 0, 0, 0.2);
}

th, td {
    padding:33px 28px 18px;
    text-align:center;
}

th {
    padding-top:22px;
    text-shadow: 1px 1px 1px #fff;
    background:#e8eaeb;
}

td {
    border-top:1px solid #e0e0e0;
    border-right:1px solid #e0e0e0;
}

tr.odd-row td {
    background:#f6f6f6;
}
```

```
td.first, th.first {
   text-align:left
}

td.last {
   border-right:none;
}

td {
   background: -moz-linear-gradient(100% 25% 90deg, #fefefe, #f9f9f9);
   background: -webkit-gradient(linear, 0% 0%, 0% 25%, from(#f9f9f9), to(#fefefe));
}

tr.odd-row td {
   background: -moz-linear-gradient(100% 25% 90deg, #f6f6f6, #f1f1f1);
   background: -webkit-gradient(linear, 0% 0%, 0% 25%, from(#f1f1f1), to(#f6f6f6));
}

th {
   background: -moz-linear-gradient(100% 20% 90deg, #e8eaeb, #ededed);
   background: -webkit-gradient(linear, 0% 0%, 0% 20%, from(#ededed), to(#e8eaeb));
}

tr:first-child th.first {
   -moz-border-radius-topleft:5px;
   -webkit-border-top-left-radius:5px; /* Saf3-4 */
}

tr:first-child th.last {
   -moz-border-radius-topright:5px;
   -webkit-border-top-right-radius:5px; /* Saf3-4 */
}

tr:last-child td.first {
   -moz-border-radius-bottomleft:5px;
   -webkit-border-bottom-left-radius:5px; /* Saf3-4 */
}

tr:last-child td.last {
   -moz-border-radius-bottomright:5px;
   -webkit-border-bottom-right-radius:5px; /* Saf3-4 */
}
```