АННОТАЦИЯ

Отчет о курсовой работе: 48 с., 16 рис., 4 табл., 2 приложения, 7 источников.

Объект исследования – процесс взаимодействия с сторонним API.

Цель работы – разработка веб-приложения для анализа популярности новостей в социальных сетях.

Метод исследования – изучение принципов разработки веб-приложений, принципов работы со сторонним API.

В работе были использованы технологии: HTML, CSS, Java Script, VK API, PyCharm, Django, Python.

В результате решения задачи было разработано веб-приложение для анализа популярности новостей в социальных сетях. Данное веб-приложение может использоваться в сфере PR, а также в научно-исследовательских целях.

Дальнейшее развитие системы связанно с расширением функционала, оптимизацией работы со сторонними API, а также увеличением списка источников информации для анализа.

HTML, CSS, JAVA SCRIPT, FRAMEWORK, BOOTSTRAP 3, PYTHON, DJANGO, MVT, API, VK API, PYCHARM, JQUERY, HIGHCHARTS

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Анализ социальных сетей стали применять во второй половине двадцатого века как дополнение к стандартному набору инструментов социальных исследователей, а сейчас это быстро развивающаяся междисциплинарная практика, которая основана на работе с данными онлайн-исследований. Одним из приоритетных направлений в данной области является анализ распространения информации, так как это характеризирует саму сеть, а также полезна в такой отрасли, как маркетинг.

Актуальность темы обусловлена современными тенденциями развития информационного общества, а также объективными реалиями существующей ситуации в различных сферах деятельности, которая характеризуется высоким уровнем недоверия общества к различным организациям.

Поэтому целью курсовой работы является разработка веб-приложения для анализа популярности новостей, распространяемых в социальных сетях. Исходя из указанной цели, можно выделить частные задачи, поставленные в курсовой работе:

1. Проанализировать существующие аналоги и их реализацию.
2. Составить техническое задание для ПО.
3. Изучить технологии: HTML, CSS, Java Script, VK API, PyCharm, Django, Python.
4. Разработать структуру программы, разработать функционал программы, соответствующий техническому заданию.
5. Разработать интерфейс и логику программы.
6. Проанализировать разработанное ПО.

1 Анализ предметной области

1.1 Состояние вопроса

Одной из главных тенденций развития Интернета последних двух-трёх лет является стремительный рост популярности социальных сетей. В последнее время социальные сети всё активнее начинают использоваться в целях продвижения того или иного субъекта или объекта. В этих условиях тема использования социальных сетей как инструмента PR становится крайне актуальной.

На рынке сервисов для анализа информации из социальных сетей уже существует много сервисов с различным функционалом, однако данных для анализа очень большее количество, поэтому рынок заполнен не в полной мере. Для сравнительного анализа были выбраны сервисы, которые выполняют функционал, заданный в курсовой работе. Среди таких были найдены «Brand Spotter – мониторинг социальных медиа» [1] и «BABKEE» [2].

Последний аналог обладает широким функционалом в целом. Для того, чтобы воспользоваться функциями, которые необходимо реализовать в курсовой работе следует:

1. Указать электронную почту и зайти на сайт по данным, отправленным на эту почту.
2. В личном кабинете создать обьект мониторинга. Данный обьект содержит фразы, по которым будут фильтроваться данные и дополнительные настройки.
3. После создания обьекта нужно подождать, пока будет произведен анализ и уже после это можно изучать результаты.

Для примера поиска была взята фраза «Павел Дуров» (имя и фамилия создателя социальной сети Вконтакте). Сервис позволяет определять авторов

и тональность записей, однако не может выдать результаты по данным, которые существовали в социальных сетях до момента создания обьекта мониторинга (рисунок 1). Поэтому сервис не пригоден для анализа данных из прошлого, при этом неудобно создавать много обьектов для мониторинга, если у пользователя появилось желание исследовать социальную сеть в целом. Данные особенности делают сервис более направленным для PR-компаний и т.д.

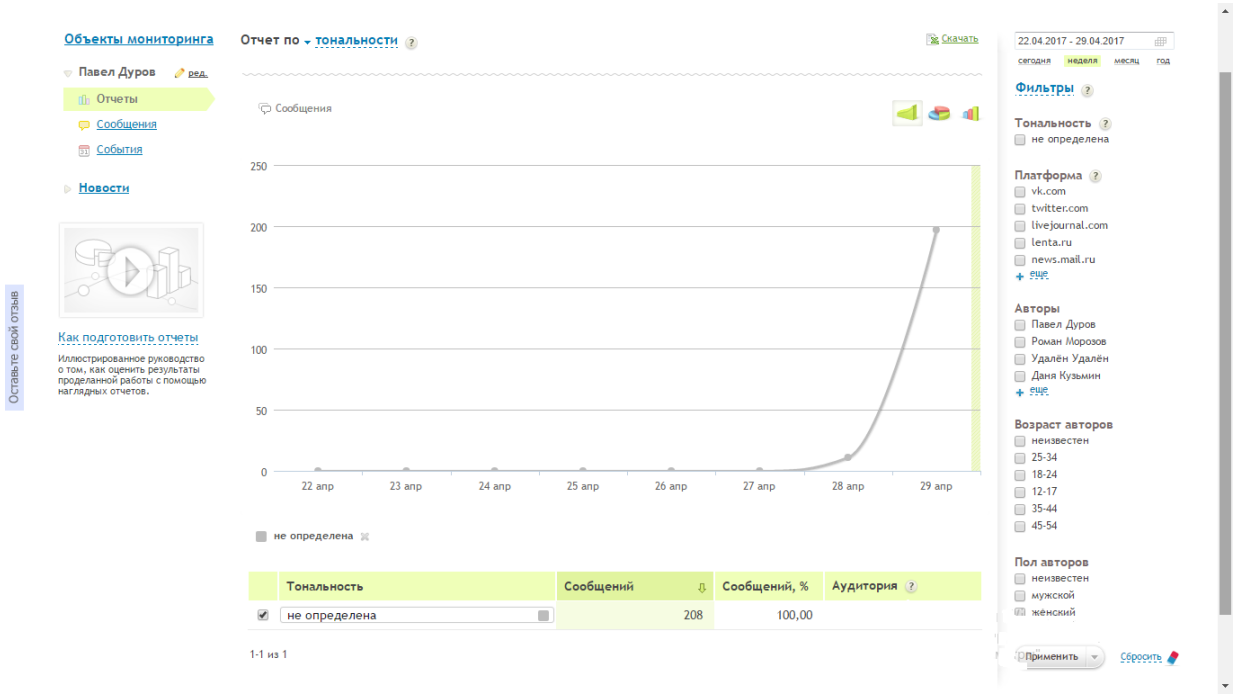

Рисунок 1 – Отчет в системе мониторинга «BABKEE»

Также, минусом данного сервиса является то, что для получения искомой информации требуется выполнить ряд действий, среди которых создание аккаунта и настройка объекта мониторинга, что может отпугнуть потенциального пользователя от данного сервиса.

Первый аналог в большей степени выполняет требуемые задачи курсовой работы, однако обладает и рядом недостатков. С результатами полного разбора сервиса можно ознакомится в таблице 1

Таблица 1 – Анализ разбора сервиса «Brand Spotter»

Название	Brand Spotter
Критерий	Анализ популярности новостей в социальных сетях
Страна	Россия
Функционал	Brand Spotter позволяет мгновенно оценить уровень репутации бренда, увидеть негативные и позитивные высказывания о компании в социальных сетях, выявить поклонников и ненавистников бренда и получить удобный и функциональный отчет.
Достоинства	1. Огромное количество разнообразного анализа (эмоциональный анализ, частотность слов в записях, список создателей записей) 2. Среди ресурсов: популярные мировые соцсети, блоги и микроблоги, главные англоязычные СМИ.
Недостатки	1. Отсутствие бесплатного использования, из-за чего допустимо только коммерческое использование. 2. Сервис не учитывает важные для социальных сетей параметры записи, как количество лайков, комментариев и репостов.

1.2 Актуальность и цель работы

Актуальность веб-сервиса для анализа популярности новостей в социальных сетях высока, так как это может быть инструментом для быстро развивающейся междисциплинарной практики исследования социальных сетей, которая основана на работе с данными онлайн-исследований, также актуальность темы обусловлена современными тенденциями развития информационного общества, а также объективными реалиями существующей ситуации в различных сферах деятельности, которая характеризуется высоким уровнем недоверия общества к различным организациям.

Внедрение разрабатываемого веб-сервиса позволит искать информацию о популярности определенной новости в социальных сетях по ключевому слову и за определенный период. Из данных, полученных в разрабатываемом веб-приложении можно будет судить о скорости роста и затухания популярности информации в социальных сетях.

Целью курсовой работы является разработка веб-приложения для анализа популярности новостей, распространяемых в социальных сетях. Исходя из указанной цели, можно выделить частные задачи, поставленные в курсовой работе:

1. Проанализировать существующие аналоги и их реализацию.
2. Составить техническое задание для ПО.
3. Изучить технологии: HTML, CSS, Java Script, VK API, PyCharm, Django, Python.
4. Разработать структуру программы, разработать функционал программы, соответствующий техническому заданию.
5. Разработать интерфейс и логику программы.
6. Проанализировать разработанное ПО.

2 Техническое задание

2.1 Общие требования к продукту

Основная задача веб-приложения – анализировать популярность информации в социальных. Более полезной будет информация за конкретный промежуток времени, поэтому, форма отправки должна содержать минимум полей: текст для поиска и промежуток времени. Данные о популярности должны выводятся пользователю в виде кривых на графике, который отображается после обработки запроса, который отправил пользователь. На графике должны отображаться следующие данные о найденных записях за определенный день:

1. Количество записей
2. Суммарное количество отметок «Мне нравится»
3. Суммарное количество репостов записей
4. Суммарное количество комментариев к записям

Для начальной разработки следует выбрать одну социальную сеть Вконтакте, поскольку данная социальная сеть одна из наиболее развитых на просторах СНГ, а также имеет специальное API для взаимодействия с ней.

2.2 Позиционирование продукта

2.2.1 Требование к пользовательским интерфейсам

Интерфейсом веб-приложения является веб-страница, отображаемая при входе на сайт. При ее создании в дизайне следует придерживаться простоты и понятности, а также текущих стандартов интерфейса веб-страниц. Разрабатываемое веб-приложение имеет специфический и небольшой функционал, поэтому для него будет достаточно одной веб-

страницы, которая будет содержать форму ввода и результирующие графики. В целом, интерфейс веб-страницы должен выглядеть, как на рисунке 2.

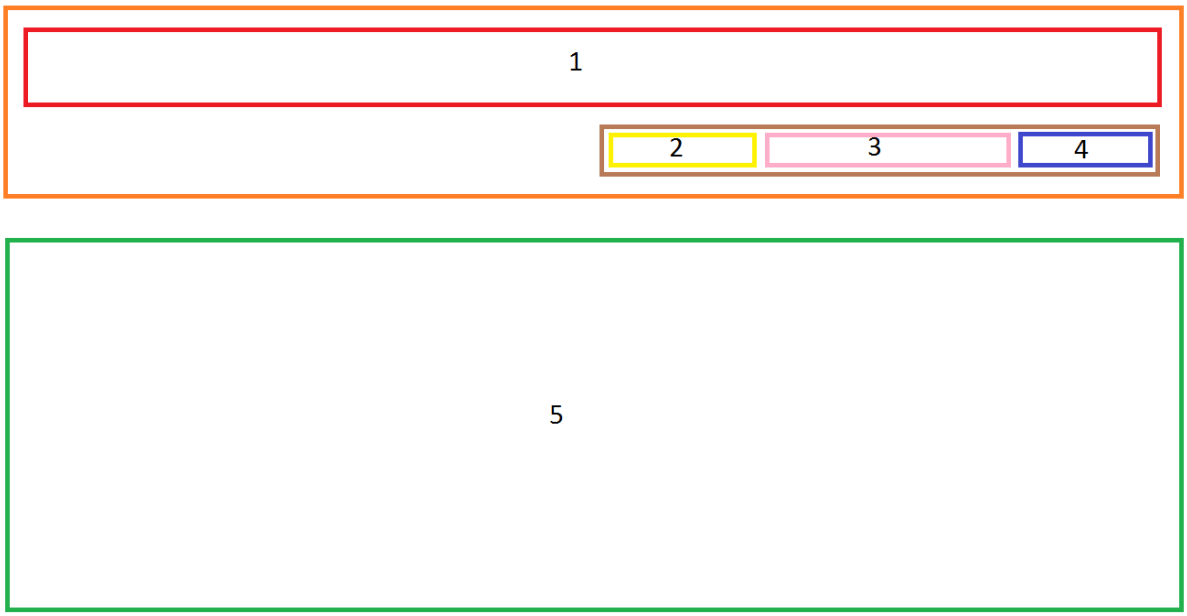

Рисунок 2 – Схема планируемого интерфейса

На рисунке 2 показаны следующие элементы интерфейса:

1. Название сервиса
2. Выбор срока для анализа
3. Элемент для ввода ключевого слова
4. Кнопка отправки
5. Результирующий график

Поскольку, график является основным обьектом веб-страницы и несет всю смысловую нагрузку, то к нему можно выдвинуть ряд следующих требований:

1. График должен отображаться только после того, как пользователь послал данные на анализ
2. Все кривые графика должны быть разных цветов или как-то отличатся друг от друга

3. Должна быть возможность масштабировать ось графика, отвечающую за количество, так как значения кривых могут значительно отличаться. Например, кол-во лайков может сильно превышать кол-во записей.

2.2.2 Требования к программным интерфейсам

Код программы должен быть понятным, содержать комментарии, не содержать повторяющегося кода и быть готовым для дальнейшей разработки. Архитектура веб-приложения должна быть реализована в соответствии со стандартами MVC.

2.2.3 Пользователи продукта и требования к ним

Разрабатываемое веб-приложение может понадобится для анализа влияния рекламы и разнообразных новостей на социальные сети, поэтому к пользователям продукта можно отнести такой круг лиц, как: студенты, менеджеры, исследователи, PR-агенты.

Опираясь на функционал веб-приложения, можно выдвинуть следующий ряд требований для вышеупомянутых пользователей:

- Подключение к сети Интернет.
- Установленный веб-браузер.

2.2.4 Функции продукта

Требуемый функционал продукта:

1. Анализ записей социальной сети Вконтакте по ключевой фразе
2. Возможность выбора промежутка времени для анализа
3. Отображение результатов анализа в виде интерактивного графика с подписями к данным

2.3 Сценарии использования продукта

Начальный сценарий: пользователь запускает браузер и открывает веб-приложение.

Сценарий 1: Открывается главная веб-страница сайта, на которой отображается название веб-приложения и однострочная форма, в которой можно выбрать период для анализа и фразу для поиска.

Сценарий 1.1: Пользователь нажимает на кнопку анализа, не вводив при этом данные для поиска. Выбивается предупреждение о том, что поле с фразой для поиска обязательно должно быть заполнено.

Сценарий 1.2: Пользователь выбирает период для анализа и вбивает новость для анализа. После нажатия на кнопку отправки страница отображается как ждущая ответа от сервера (на шапке вкладки отображается соответствующая анимация). Дале, после получения ответа от сервера страница перезагружается и под формой на всю страницу отображается график с кривыми. Даже после перезагрузки страницы в поле поиска остается фраза для анализа.

Сценарий 1.2.1: При наведении на любую точку графика должна отображаться вспомогательная табличка с обозначением данных, которые отображает данная точка.

Сценарий 1.2.2: При щелчке на обозначение любой из кривых графика данная кривая исчезает из графика, а график масштабируется под оставшиеся кривые.

3. Проектирование программного продукта

3.1 Обоснование средств разработки

3.1.1 HTML5

HTML5 (англ. HyperText Markup Language, version 5) — язык для структурирования и представления содержимого всемирной паутины. Это пятая версия HTML. Хотя стандарт был завершён (рекомендованная версия к использованию) только в 2014 году (предыдущая, четвёртая, версия опубликована в 1999 году), ещё с 2013 года браузерами оперативно осуществлялась поддержка, а разработчиками — использование рабочего стандарта (англ. HTML Living Standard). Цель разработки HTML5-улучшение уровня поддержки мультимедиа-технологий с одновременным сохранением обратной совместимости, удобочитаемости кода для человека и простоты анализа для парсеров.

В HTML5 реализовано множество новых синтаксических особенностей. Например, элементы <video>, <audio> и <canvas>, а также возможность использования SVG и математических формул. Эти новшества разработаны для упрощения создания и управления графическими и мультимедийными объектами в сети без необходимости использования сторонних API и плагинов. Другие новые элементы, такие как <section>, <article>, <header> и <nav>, разработаны для того, чтобы обогащать семантическое содержимое документа (страницы). Новые атрибуты были введены с той же целью, хотя ряд элементов и атрибутов был удалён. Некоторые элементы, например, <a>, <menu> и <cite>, были изменены, переопределены или стандартизированы.

3.1.2 CSS3

CSS3 (англ. Cascading Style Sheets 3 — каскадные таблицы стилей третьего поколения) — активно разрабатываемая спецификация CSS. Представляет собой формальный язык, реализованный с помощью языка разметки. Самая масштабная редакция по сравнению с CSS1, CSS2 и CSS2.1. Главной особенностью CSS3 является возможность создавать анимированные элементы без использования JS, поддержка линейных и радиальных градиентов, теней, сглаживания и многое другое.

CSS3 преимущественно используется как средство описания и оформления внешнего вида веб-страниц, написанных с помощью языков разметки HTML и XHTML, но может также применяться к любым XML-документам, например, к SVG или XUL.

В отличие от предыдущих версий спецификация разбита на модули, разработка и развитие которых идёт независимо. CSS3 основан на CSS2.1, дополняет существующие свойства и значения и добавляет новые нововведения, начиная с малых, вроде закругленных углов блоков, заканчивая трансформацией (анимацией) и, возможно, введением переменных.

3.1.3 JavaScript

JavaScript — прототипно-ориентированный сценарный язык программирования. Является реализацией языка ECMAScript (стандарт ECMA-262).

JavaScript обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам.

На JavaScript оказали влияние многие языки, при разработке была цель сделать язык похожим на Java, но при этом лёгким для использования непрограммистами. Языком JavaScript не владеет какая-либо компания или организация, что отличает его от ряда языков программирования, используемых в веб-разработке.

3.1.4 Twitter Bootstrap

Bootstrap 3 предоставляет набор интрументов для создания сайтов и веб-страниц. Данный фреймворк включает в себя HTML и CSS шаблоны, а также включает в себя расширения Javascript. Bootstrap позволяет создать адаптивный сайт с помощью резиновой системы разметки, которая масштабируется до 12 столбцов на различных устройсвах. Подобный масштаб позволяет создавать как простые варианты разметки, так и более сложные макеты. Также, данный фреймворк учитывает новые стандарты дизайна пользовательского интерфейса и правила оформления веб-страниц.

3.1.5 Django framework

Django (Джанго) — свободный фреймворк для веб-приложений на языке Python, использующий шаблон проектирования MVC. 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.

3.1.6 PyCharm IDE

PyCharm — интегрированная среда разработки для языка программирования Python. Предоставляет средства для анализа кода, графический отладчик, инструмент для запуска юнит-тестов. PyCharm разработана компанией JetBrains на основе IntelliJ IDEA.

Основным критерием выбора данной IDE является то, что она поддерживает веб-разработку на Django. Среди возможностей PyCharm, помогающих в разработке проекта можно выделить следующие:

- Создание каркаса проекта Django
- Запуск тестового сервера Django, нажатием одной кнопки
- Отображение лога сервера в интегрированной в IDE консоли
- Быстрый запуск командной строки
- Подсветка синтаксиса, в том числе и синтаксиса шаблонов Django

3.2 Описание используемых API

API (application programming interface) — это посредник между разработчиком приложений и какой-либо средой, с которой это приложение должно взаимодействовать. API упрощает создание кода, поскольку предоставляет набор готовых классов, функций или структур для работы с имеющимися данными.

API ВКонтакте (Vk API) — это интерфейс, который позволяет получать информацию из базы данных vk.com с помощью http-запросов к специальному серверу. Разработчику не нужно знать в подробностях, как

устроена база, из каких таблиц и полей каких типов она состоит — достаточно того, что API-запрос об этом «знает». Синтаксис запросов и тип возвращаемых ими данных строго определены на стороне самого сервиса.

Например, для получения данных о пользователе с идентификатором 210700286 необходимо составить запрос вида, как на рисунке 3.

```
https://api.vk.com/method/users.get?user_id=210700286&v=5.52
```

Рисунок 3 – Пример получения данных о пользователе
с помощью запроса к VK API

Рассмотрим отдельно все составляющие запроса из рисунка 7:

- https:// — протокол соединения
- api.vk.com/method — адрес API-сервиса
- users.get — название метода API ВКонтакте. Методы представляют собой условные команды, которые соответствуют некоторой операции с базой данных — получение информации, запись или удаление
- ?user_id=210700286&v=5.52 — параметры запроса. После названия метода нужно передать его входные данные (если они есть) — как обычные GET-параметры в http-запросе. В нашем примере мы сообщаем серверу, что хотим получить данные о пользователе с id=210700286 и формат этих данных должен соответствовать версии API 5.52. Входные параметры всегда перечислены на странице с описанием метода

В ответ сервер вернет JSON-объект с запрошенными данными (или сообщение об ошибке, если что-то пошло не так). JSON — это формат записи данных в виде пар «имя свойства»: «значение». Ответ на запрос из примера (рисунок 3) выглядит, как на рисунке 4:

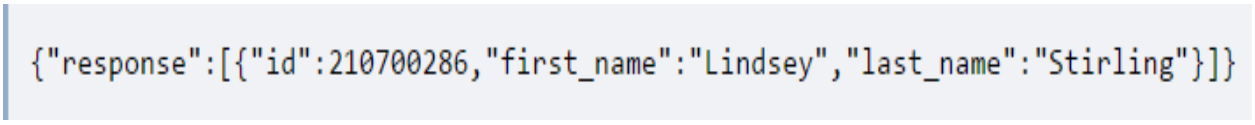

Рисунок 4 – Пример ответа от VK API

Структура ответа каждого метода также строго задана, и при работе с API Вы заранее знаете, что в поле id придет число, а в поле first_name — строка. Такие правила оговариваются на страницах с описанием метода и соответствующих объектов, которые он возвращает в ответе. Например, users.get — здесь описаны входные параметры метода и структура его ответа, а здесь — user подробно расписано каждое поле объекта из ответа.

Объект из ответа может быть не уникален для конкретного метода. Например, объект пользователя с набором полей, содержащих данные о его образовании, возрасте, интересах, может возвращаться в ответе от методов users.get, users.search, groups.getMembers и еще нескольких.

Также, у API Вконтакте есть ограничения на использование их методов. В общем, ограничения делятся на частотные и количественные. Частотные ограничения заключаются в том, что к методам API ВКонтакте (за исключением методов из секций secure и ads) можно обращаться не чаще 3 раз в секунду. В количественных ограничениях руководство Вконтакте не может сказать точные ограничения в силу того, что злоумышленники могут это использовать, однако известно, что после превышения количественного лимита доступ к конкретному методу может требовать ввода captcha, а также может быть временно ограничен (в таком случае сервер не возвращает ответ на вызов конкретного метода, но без проблем обрабатывает любые другие запросы).

К ограничениям запросов можно еще отнести ограничение на количество возвращаемых данных, поскольку некоторые из запросов могли бы вернуть огромные количества данных. Данные ограничения относятся только к тем методам, которые возвращают данные, например, к методу

newsfeed.search, который возвращает записи пользователей и групп по определенному запросу. Данный метод не может вернуть более 200 записей за один запрос, а также более 1000 записей по искомым параметрам, что следует учесть в курсовой работе.

3.3 Проектирование алгоритмов приложения

Перед началом создания веб-приложения следует создать концептуальные алгоритмы работы приложения, которые бы небыли привязаны к аппаратным особенностям и технологиям, кроме как к понятию веб-сервера.

Изображенная на рисунке 5 схема описывает взаимодействие клиента и сервера, а также описывает данные, которыми они оперируют и содержит алгоритм получения данных от социальной сети Вконтакте. Данная схема была создана с помощью ресурса draw.io [3].

Рисунок 5 – Концептуальная схема
взаимодействия веб-приложения и клиента

4. Разработка ПО

4.1 Создание проекта веб-приложения

Для разработки веб-приложения был выбран фреймворк Django, поэтому для создания веб-приложения необходимо начать с создания проекта Django. Использование PyCharm IDE при разработке ПО значительно сократить выполнение анной задачи. Для этого нужно вызвать окно диалога для создания нового проекта, затем выбрать тип проекта (Django). Далее следует выполнить следующие шаги:

- Выбрать каталог для создания проекта
- Выбрать интерпретатор Python (если на нем не будет установлен Django, то он автоматически установится)
- Выбрать название общей папки для хранения шаблонов
- Выбрать название приложения (если не будет введено, то проект будет создан без приложения)

После заполнения необходимых полей для разрабатываемого проекта окно диалога будет выглядеть, как на рисунке 6.

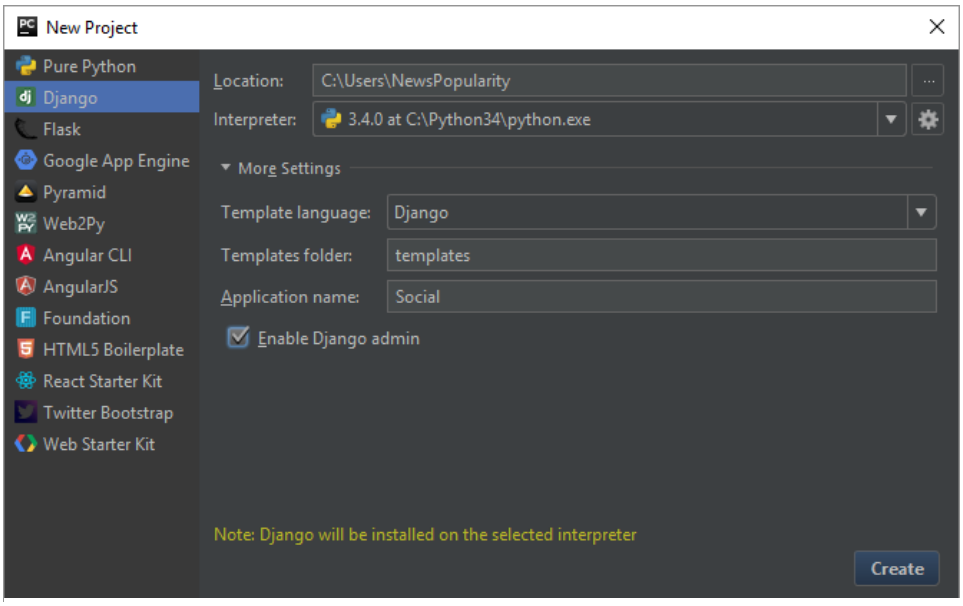

Рисунок 6 – Диалог создание проекта Django

После выполнения вышеуказанных действий будет создан проект Django, содержащий папку с приложением, в которой будут содержатся основные файлы проекта.

Django фреймворк использует свой аналог концепции MVC, под названием MVT (model-template-view). Это значит, что, при отсутствии базы данных в проекте, следует создать еще шаблоны (templates) и представления (views).

Первое, что нужно сделать после создания проекта – это настроить файлы urls.py. Этот модуль содержит код Python, который отображает URL-шаблоны (регулярные выражения) и связанные функции Python (представления). Этот модуль может быть короткий или длинный настолько, насколько это нужно. Он может ссылаться на другие такие модули. И, так как это код Python, может создаваться динамически.

4.2 Создание контроллера

Для шаблона, который будет превращаться в html-страницу, нужны данные. В Django-проекте они передаются в шаблон с помощью view (вид, представление, аналог контроллера в концепции MVC). Поскольку, шаблон веб-страницы еще не создан, будем полагать, что данные для анализа поступают в форме с именами days (количество дней для анализа) и text (фраза для фильтрации записей).

Данные в представление передаются в обьекте-словаре, как один из параметров метода. Для того, чтобы достать эти данные можно использовать стандартные методы извлечения данных из словаря по ключу в Python 3.4. Например, метод get или обращению по названию ключа. В целом, инициализация необходимых переменных для дальнейшего анализа выглядит, как на рисунке 7.

```
# количество дней для анализа
days_delta = int(request.POST['days'])
# Дата начала
start_date = datetime.now() - timedelta(days=days_delta)
# Конец анализа наступает в данный момент
end_date = datetime.now()
# текст поиска
text = request.POST['text']
# в начальной дате отбрасываются часы, минуты. . .
temp_date = start_date.replace(hour=0, minute=0, second=0, microsecond=0)
# создание обьекта для последующего заполнения и передачи его в шаблон
data = {'likes': [], 'reposts': [], 'comments': [], 'posts': [], }
```

Рисунок 7 – Работа с обьектом POST запроса в Python

Данные из формы можно получить только, если пользователь их передал. В Django framework при отправке форм всегда используется только метод POST, поэтому можно отследить, передал ли пользователь данные с помощью свойства method обьекта-запроса, переданного во view. Пример определения метода запроса к серверу можно увидеть на рисунке 8.

```
if request.method == 'POST':
    # используется метод POST
```

Рисунок 8 – Определение типа запроса в Django

4.3 Работа с API Вконтакте

Для вызова методов VK API требуется выполнение https-запросов по специальным адресам и со специальными параметрами, поэтому, для облегчения работы с API было решено использовать стороннюю Python-библиотеку vk. Работа с данной библиотекой значительно облегчает использование API Вконтакте, до уровня вызова его методов в коде скрипта с необходимыми параметрами, а для начала работы сводится до создания сессии и подключения, как показано на рисунке 9.

```
# активируем vk.API
session = vk.AuthSession(access_token=secret_key)
api = vk.API(session)
```

Рисунок 9 – Подключение к API Вконтакте в Python

Secret_key (рисунок 9) - это переменная, которая содержит секретный ключ приложения, которое нужно создать. Для создания приложения Вконтакте следует:

1. Перейти на сайт разработчиков Вконтакте [4]
2. Зайти в меню «Мои приложения», и затем нажать кнопку «Создать приложение»
3. Выбрать название и тип приложения (Standalone-приложение)
4. Настроить оставшиеся настройки

После успешного создания приложения можно будет вставить его в код приложения. При этом, если секретный ключ будет скомпрометирован, то можно будет безопасно его обновить.

Для поиска новости в социальной сети Вконтакте был использован метод newsfeed.search. Новости возвращаются в порядке от более новых к более старым, однако есть ограничения на отправку запросов и получение данных, описанные в главе 3.2. Для решения данной проблемы был разработан следующий алгоритм:

1. Отправляется запрос с параметрами времени начала и окончания дня, версией API, текстом и значением количества записей для получения
2. Получаем данные, переносим новости в массив всех новостей за день для дальнейшего анализа
3. Если в ответе на запрос есть параметр next_form, то посылаем еще запрос, в котором устанавливаем параметр start_form, равный полученному параметру next_form и повторяем шаг 2

4.4 Создание пользовательского интерфейса

После создания представления следует создать и шаблон веб-страницы, который будет динамически создаваться в зависимости от полученных данных от представлений и отправлятся обратно клиенту, что и будет интерфейсом разработываемого ПО. Данный шаблон состоит из стандартной html-разметки и шаблонных тегов фреймворк Django.

4.4.1 Html-разметка шаблонов

Для создания html-разметки сайта использовался фреймворк Twitter Bootstrap. Он предоставляет набор интрументов для создания сайтов и различных веб-страниц. Данный фреймворк включает в себя HTML и CSS шаблоны, а также включает в себя расширения Javascript.

Bootstrap позволяет создать адаптивные веб-страницы с помощью резиновой системы разметки, которая масштабируется до 12 столбцов на различных устройсвах. Подобный масштаб позволяет создавать как простые варианты разметки, так и более сложные макеты. Для верстки сайта использовались классы, описанные в таблице 2.

Таблица 2 – Используемые css-классы при верстке

Класс	Описание
Container	Блок-обертка всего содержимого, задает основную ширину сайта
Row	Базовый блок для колонок
Jumbotron	Класс предназначен для создания контента или информации на веб-сайте, который создан привлечь внимание пользователя
Form-inline	Класс для создания однострочной формы

В таблице 2 отображены основные классы, которые неоднократно использовались в проекте. По мимо данных, использовались и дополнительные классы, которые помогают позиционировать текст в блоках, прятать блоки, выделять текст особым цветом и так далее. Также, основные классы Bootstrap 3 используют вложенные подклассы для разделения содержимого блоков.

С использованием классов Bootstrap была создана html-разметка стартового представления страницы (без графика), как на рисунке 10.

```
<div class="container">
    <div class="jumbotron">
        <h1>Анализ новостей в социальной сети <a href="https://vk.com">Вконтакте</a></h1>
        <h4>
            <form class="form-inline text-right" action="" method="POST">
                {% csrf_token %}
                <div class="form-group">
                    <blockquote class="blockquote-reverse">
                        <label for="days">Период</label>
                        <select class="form-control" id="days" name="days">
                            <option value="3">Три дня</option>
                            <option value="7">Неделя</option>
                            <option value="14">Две недели</option>
                            <option value="30">30 дней</option>
                            <option value="360">Год</option>
                        </select>
                        <input class="form-control" type="text" name="text" placeholder="Новость для поиска. . ."
                               required {% if text %}value="{{ text }}"{% endif %}>
                        <input type="submit" class="btn btn-primary" value="Отправить">
                        <p></p>
                        <footer>Каждый день обрабатывается ~1 сек.</footer>
                    </blockquote>
                </div>
            </form>
        </h4>
    </div>
```

Рисунок 10 – Разметка стартовой веб-страницы

На рисунке 10 атрибут required обязывает пользователя ввести данные в поле текстового ввода перед отправки формы, а атрибуты value тега select отвечают за значение, которое будет послано в форме. Также, был использован шаблонный тег if, который будет рассмотрен в следующей части главы.

4.4.2 Отображение графика

Для отображения графика была использована Java Script библиотека Highcharts. Highcharts – это бесплатный некоммерческий проект визуализации, который распространяется бесплатно. В данный момент поддерживает различные типы диаграмм (окружные, столбиковые и т.д.). Highcharts совместим со всеми браузерами, он будет работать даже в IE6. Далее представлены некоторые характеристики Highcharts:

- Поддержка многих типов графиков и диаграмм
- Легкая настройка посредством JSON (javascript Object Notation)
- Всплывающая подсказка с отображением информации для каждого отрезка
- Поддержка масштабирования
- Возможность подгружать внешние данные

Таким образом, воспользовавшись официальной документацией Highcharts [5] и примерами графиков был создан основной график приложения, листинг которого можно посмотреть в приложении Б.1. Логически его можно поделить на три части. Первая часть, это просто тег с определенным идентификатором, в который позже методы из библиотеки Highcharts вставят график. Вторая часть и третья части –это Java Script код. Вторая логическая часть создания графика – это установка русского языка в график, так как по умолчанию используется английский язык. Последняя часть и основная – создание обьекта, который представляет график. Среди его свойств в разрабатываемом веб-приложении устанавливается:

- Заголовок и подзаголовок
- Подписи к осям
- Подписи к точкам графика
- Позиционирование названия кривых и их выравнивание
- Массивы с данными в качестве данных для кривых

4.4.3 Использование шаблонов Django

Перед отправкой веб-страницы пользователю она компилируется специальной подпрограммой Django, что позволяет использовать «шаблонные теги и фильтры». Данные теги помогают сделать страницу более динамичной, так как есть возможность подставить туда данные из представления. Для разработки понадобились следующие шаблонные теги:

- Load – загрузка библиотеки тегов
- Csrf_token - в Django 1.2 и выше, используется для CSRF защиты
- Static - создает ссылку на файл в директории со статическими файлами
- For – создает цикл при компиляции шаблона по итерируемому обьекту
- If – шаблонный тег условного выполнения

Чтобы воспользоваться шаблонным тегом, необходимо указать его имя и его параметры через пробел между «{%» и «%}» прямо внутри разметки шаблона. Первым, используется тег load с параметром static, что позволит использовать последний в шаблоне. Шаблонный тег static используется для указания пути к статическим файлам в тегах link и script. Тег csrf_token доступен и без подключения, и для его использования данный шаблонный тег нужно использовать в любом месте тега form.

На рисунке 10 можно ознакомится с примером использования условного шаблонного тега if. Он срабатывает так, что блок кода между {% if %} и {% endif %} выполнится, если переменная text передана в шаблон. Подобным образом скрывается график при GET запросе. Т.е. блок кода с графиком не выполняется, если в шаблон не переданы данные для графика.

Шаблонный тег for выполняет основные действия шаблонизатора – заполняет скрипт данными, пример использования такого тега можно увидеть на рисунке 11.

```
series: [{
    name: 'Лайки',
    data: [{% for item in data.likes %}
        [{{ item.0 }}, {{ item.1 }}],
    {% endfor %}]
}, {
    name: 'Репосты',
    data: [{% for item in data.reposts %}
        [{{ item.0 }}, {{ item.1 }}],
    {% endfor %}]
}, {
    name: 'Комментарии',
    data: [{% for item in data.comments %}
        [{{ item.0 }}, {{ item.1 }}],
    {% endfor %}]
}, {
    name: 'Записи',
    data: [{% for item in data.posts %}
        [{{ item.0 }}, {{ item.1 }}],
    {% endfor %}]
}]
```

Рисунок 11 – Пример использования тега for

Четыре однотипных шаблонных тега for проходят по объектам массивов likes, reposts, comments и posts и подставляют их в генерируемую веб-страницу, в результате чего получается js-код в html-разметке, пример которого можно увидеть на рисунке 12.

```
series: [{
    name: 'Лайки',
    data: [
        [Date.UTC(2017,3,24), 1491],

        [Date.UTC(2017,3,25), 856],

        [Date.UTC(2017,3,26), 1412],

        [Date.UTC(2017,3,27), 2477],

        [Date.UTC(2017,3,28), 1448],

        [Date.UTC(2017,3,29), 443],

        [Date.UTC(2017,3,30), 2004],

        [Date.UTC(2017,4,1), 0],
    ]
}, {
    name: 'Репосты'
```

Рисунок 12 – Пример генерируемой html-разметки

5. Описание программного продукта

5.1 Структура проекта

Файловая структура является структурой проекта, а также структурой модулей Python, что является важной составляющей Django-проекта. Файловую структуру проекта можно рассмотреть на рисунке 13.

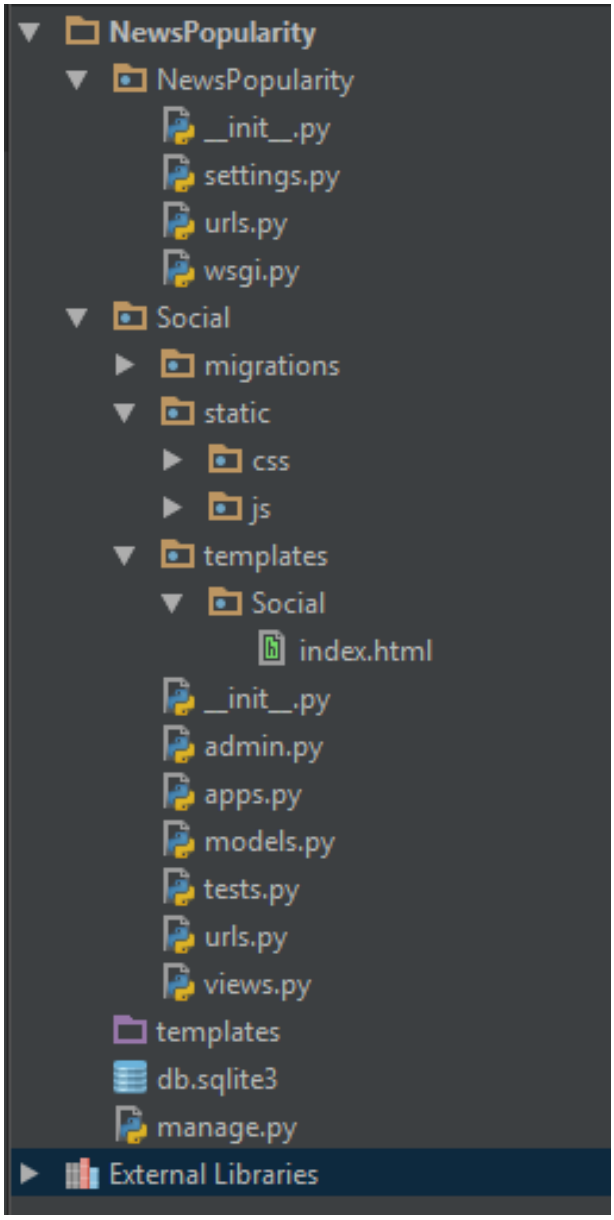

Рисунок 13 – Файловая структура проекта

Данная структура проекта (рисунок 13) является стандартной структурой Django-проектов и достигается с помощью выполнения команд startproject и startapp, однако данные команды не пришлось выполнять вручную, так как среда разработки PyCharm выполнила их автоматически. На первом шаге, с помощью пакета Django была создана следующая структура:

- Корневая директория NewsPopularity/: – это корневая директория проекта. Его название никак не используется Django, можно переименовать его как угодно
- manage.py: Скрипт, который позволяет взаимодействовать с проектом Django
- Внутренний каталог NewsPopularity/ - это пакет Python проекта. Его название – это название пакета Python, которое используется для импорта чего-либо из проекта (например, Site.urls)
- NewsPopularity/__init__.py: Пустой файл, который указывает Python, что текущий каталог является пакетом Python
- NewsPopularity/settings.py: Настройки/конфигурация проекта
- Site/urls.py: Конфигурация URL-ов для проекта Django. Это “содержание” всех Django-сайтов
- NewsPopularity/wsgi.py: Точка входа проекта для WSGI-совместимых веб-серверов

На следующем шаге, командой startapp, был создан каталог внутри корневой папки с файлами проекта со следующим содержимым:

- __init__.py: Пустой файл, который указывает Python, что текущий каталог является пакетом Python
- migrations/: Директория, в которой сохраняются миграции
- static/: Папка со статическими файлами приложения (скриптами, файлами стилей и т.д.)
- templates/: Папка с шаблонами
- apps.py: Модуль, который добавляет в проект текущее приложение

Помимо автоматически созданных файлов, в директорию с приложением была добавлена директория static для статических фалов стилей и Java Script файлов, которые разделены по разным поддиректориям.

Также, в папке с приложением была создана папка templates для шаблонов. Данная директория должна иметь специальную структуру, в которой шаблоны, относящиеся к приложению должны содержаться в папке, которая имеет такое же название, как и приложение.

5.2 Описание объектов и их взаимодействия

Архитектура приложения зачастую зависит от архитектуры фреймворка, с помощью которого он был создан. Django-приложение состоит из четырех основных компонентов: модель данных, представление, шаблоны, URL.

Любая модель является стандартным python-классом. Объектно-ориентированный ORM обеспечивает таким классам доступ непосредственно к базам данных. Если бы не было ORM, программисту пришлось бы писать запросы непосредственно на SQL, однако в данном проекте не было необходимости создавать свои модели.

Представление - объекты типа view в django выполняют разнообразные функции, в том числе контролируют запросы пользователя, выдают контекст в зависимости от его роли. View – это обычная функция, которая вызывается в ответ на запрос какого-то адреса (url) и возвращает контекст. В данном проекте существует только одно представление index.

Шаблоны являются формой представления данных. Шаблоны имеют свой собственный простой метаязык и являются одним из основных средств вывода на экран.

URL - это всего лишь механизм внешнего доступа к представлениям (view). Встроенные в url-файлы регулярные выражения делают механизм

достаточно гибким. При этом одно представление может быть сконфигурировано к нескольким url-файлами, предоставляя доступ различным приложениям. Здесь поддерживается философия закладок: url-файлы становятся как бы самодостаточными и начинают жить независимо от представления.

Общая схема взаимодействия компонентов любого Django-проекта показана на рисунке 14.

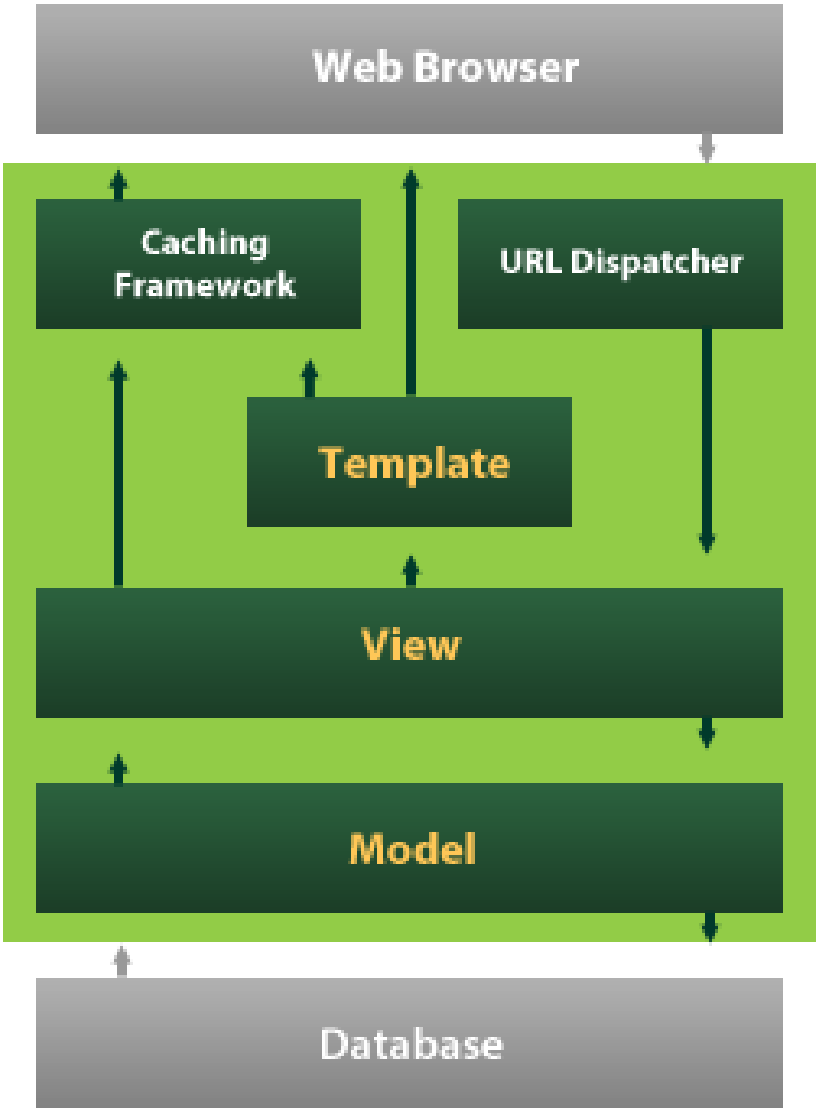

Рисунок 14 – Архитектура Django-приложения

6. Тестирование и внедрение

6.1 Тестирование разработанного ПО

Назначение, область применения и требования к надежности программного продукта показаны в таблице 3, а результаты анализа надежности в таблице 4.

Таблица 3 – Назначение, область применения и требования к надежности

Назначение	Область применения	Требования к надежности
Анализ популярности информации в социальных сетях	Маркетинг, исследования	Скорость обработки, достоверность

Таблица 4 - Анализ надежности программного продукта

№ Теста	Данные для теста (вводимая информация)	Время функционирования программы	Результат (успешно\ отказ)	Результат
1.	Открытие сайта	Менее секунды	успешно	Отображение веб-страницы сервиса
2.	Отправка запроса для анализа с параметрами (период – 7 дней, текст – Павел Дуров)	6.65 сек	успешно	Перезагрузка страницы с появлением графика за 7 дней

Окончание таблицы 4

3.	Отправка запроса для анализа с параметрами (период – 14 дней, текст – Павел Дуров)	12.3 сек	успешно	Перезагрузка страницы с появлением графика за 14 дней
4.	Отправка запроса для анализа без текста для анализа	Моментально	отказ	Появление предупреждения о том, что поле с текстом должно быть заполнено
5.	Нажатие на подпись к кривой на графике	Моментально	успешно	Исчезновение кривой, обновление масштабов графика

6.2 Установка программы

6.2.1 Необходимые аппаратные и программные средства

Для установки ПО действуют следующие системные требования:

- Не менее 500 mb памяти на жестком диске
- Процессор – Intel Pentium CPU 2117U@ 1.80GHz (2 ядра)
- Видеокарта от 256 mb памяти
- Оперативная память от 1 GB и выше
- Доступ к сети Internet

Поскольку, Python мультиплатформенный, то данное веб-приложение может работать практически на любой системе, на которой будет работать Python и будет доступ к интернету.

Для функционирования ПО действуют следующие минимальное программные требования:

- Django v.1.10
- Vk v.2.0.2
- Python v.3.4

Для полноценной работы веб-приложения необходимо произвести развертывание проекта на полноценный веб-сервер (Apache, например).

6.2.2 Описание процесса развертывания

Основной платформой для развертывания Django является WSGI, это фактически стандарт для веб-серверов и приложений на Python. Команда startproject установит простую WSGI конфигурацию по умолчанию, которую можно впоследствии изменить под нужды проекта и использовать с любым WSGI-совместимым веб-сервером.

Одна из ключевых концепций развертывания с WSGI заключается в указании функции, или вызываемого объекта, application, который использует веб-сервер для взаимодействия с кодом. Обычно это объект application модуля Python доступного для сервера.

Команда startproject создаст файл <project_name>/wsgi.py, который содержит вызываемый объект application. Этот файл использует как встроенным сервером для разработки, так и на боевом WSGI сервере.

WSGI сервер получает путь к объекту application из своих настроек. Встроенный сервер Django использует настройку WSGI_APPLICATION. По умолчанию она равна <project_name>.wsgi.application, и указывает на объект application в <project_name>/wsgi.py.

Когда WSGI сервер загружает ваше приложение, Django необходимо импортировать модуль с настройками. Django использует переменную окружения DJANGO_SETTINGS_MODULE для определения расположения модуля настроек. Она должна содержать путь для импорта этого модуля. Если переменная не определена, wsgi.py использует значение mysite.settings, где mysite название вашего проекта.

В данной курсовой работе будет рассмотрен самый популярный вариант развёртывания - Apache и mod_wsgi. Mod_wsgi является модулем веб-сервера Apache, который может взаимодействовать с любым приложением Python, в том числе Django. Django работает с любой версией Apache, поддерживающей mod_wsgi.

После установки и активации mod_wsgi нужно отредактировать файл httpd.conf веб-сервера Apache, изменив его определенным образом. Значение WSGIScriptAlias указывает местоположение приложений, (/ обозначает корневую директорию), вторым значением указывается расположение файла “WSGI (как правило, в корне проекта). Эти настройки позволят Apache обрабатывать любой запрос из директории, указанной как базовая с помощью WSGI-приложения, хранящегося в ней. WSGIPythonPath гарантирует, что проект доступен для импорта; иначе говоря, что команда import Site сработает. Значение <Directory> просто предоставляет Apache доступ к файлу wsgi.py.

ЗАКЛЮЧЕНИЕ

В результате работы над курсовой работой выполнены все поставленные задачи. Разработано работоспособное веб-приложение, способное производить анализ популярности информации в социальной сети Вконтакте. Проанализированы существующие аналоги, выделены базовые критерии оценки. Был изучен ряд технологий для создания проекта: HTML, CSS, Java Script, VK API, PyCharm, Django, Python. Также, было написано обоснование выбора данных технологий в курсовом проекте. Функционал веб-приложения был утвержден согласно разработанной структуре программы. На основе разработанной функциональной схемы проекта составлен программный продукт, обладающий серверной и интерфейсной частью и работой со сторонними API.

На данный момент при работе с веб-приложением доступны следующие возможности: выбор периода для анализа, выбор ключевой фразы для анализа, работа с интерактивным графиком.

Дальнейшее развитие программы связано с расширением ее возможностей по анализу, добавлением новых социальных сетей и оптимизацией скорости работы. Тестирование программы показало ее работоспособность и высокую степень надежности.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Brand Spotter – мониторинг социальных медиа:

URL: https://brandspotter.ru/ (дата обращения: 18.03.2017)

2. Babkee - мониторинг и исследования медиа:

URL: http://www.babkee.ru/ (дата обращения: 18.03.2017)

3. Draw.io – онлайн конструктор диаграм:

URL: https://www.draw.io/ (дата обращения: 25.03.2017)

4. Официальная страница фреймворка Bootstrap 3

URL: http://getbootstrap.com/ (дата обращения: 26.03.2017)

5. Wikipedia – информационный ресурс:

URL: https://ru.wikipedia.org/wiki/HTML5

(дата обращения: 15.04.2017)

URL: https://ru.wikipedia.org/wiki/JavaScript

(дата обращения: 15.04.2017)

URL: https://en.wikipedia.org/wiki/Brackets_(text_editor)

(дата обращения: 16.04.2017)

6. Документация API Вконтакте

URL: https://vk.com/dev/manuals (дата обращения: 19.04.2017)

7. w3school: информационный ресурс

URL: https://www.w3schools.com/html/default.asp

(дата обращения: 21.02.2017)

URL: https://www.w3schools.com/css/default.asp

(дата обращения: 22.02.2017)

URL: https://www.w3schools.com/js/default.asp

(дата обращения: 25. 02.2017)

ПРИЛОЖЕНИЕ А

(обязательное)

Экранные формы

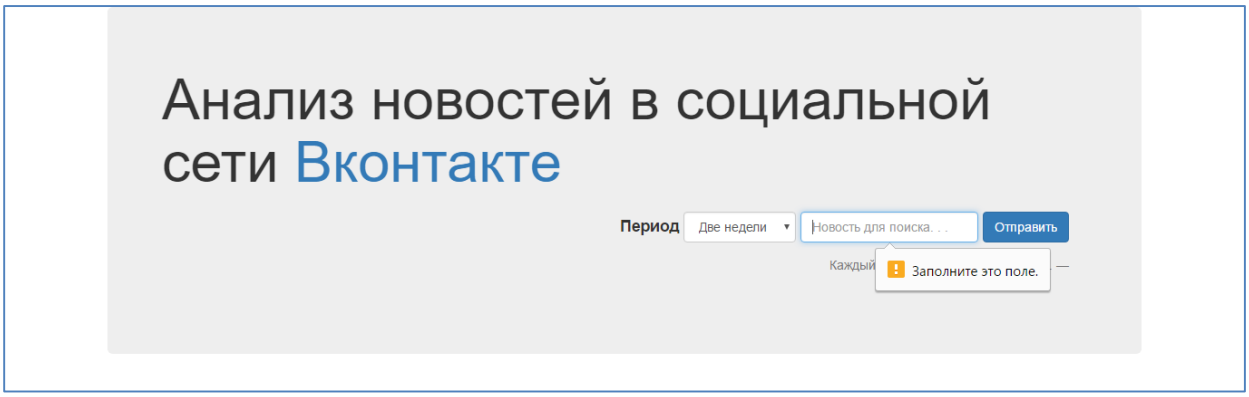

Рисунок А.1 – Результат отправки запроса
с незаполненной формой

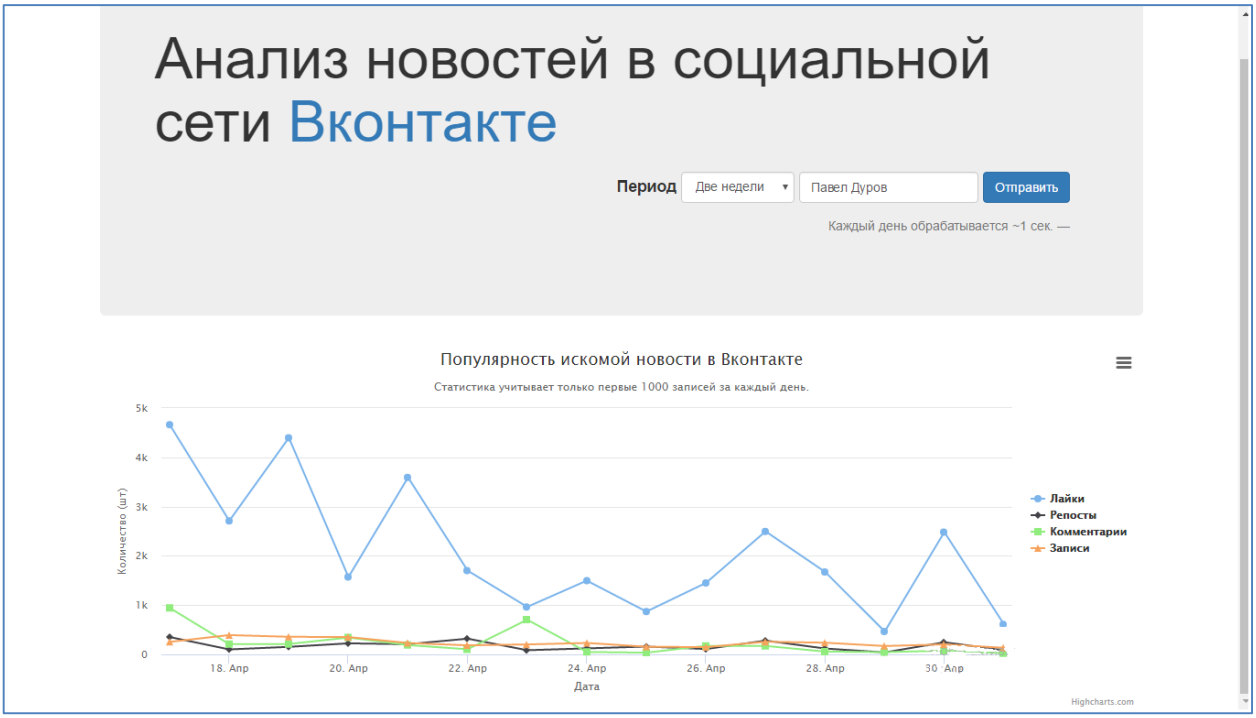

Рисунок А.2 – Пример отображения результатов анализа

ПРИЛОЖЕНИЕ Б

(обязательное)

Фрагменты листинга

Листинг Б.1 Файл index.html

```
{% load staticfiles %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>

    <!-- MaxCDN: Bootstrap 3 CSS -->
    <script src="{% static 'js/jquery.min.js' %}"></script>
    <script src="{% static 'js/moment.js' %}"></script>
    <script src="{% static 'js/bootstrap.min.js' %}"></script>
    <script src="{% static 'js/bootstrap-datepicker.js' %}"></script>

    <script src="https://code.highcharts.com/highcharts.src.js"></script>
    <script src="https://code.highcharts.com/modules/exporting.js"></script>
    <link rel="stylesheet" href="{% static 'css/bootstrap.css' %}">
    <link rel="stylesheet" href="{% static 'css/bootstrap-datepicker3.css' %}">
</head>
<body>
```

```
<div class="container">
    <div class="jumbotron">
        <h1>Анализ новостей в социальной сети <a href="https://vk.com">Вконтакте</a></h1>
        <h4>
            <form class="form-inline text-right" action="" method="POST">
                {% csrf_token %}
                <div class="form-group">
                    <blockquote class="blockquote-reverse">
                        <label for="days">Период</label>
                        <select class="form-control" id="days" name="days">
                            <option value="3">Три дня</option>
                            <option value="7">Неделя</option>
                            <option value="14">Две недели</option>
                            <option value="30">30 дней</option>
                            <option value="360">Год</option>
                        </select>
                        <input class="form-control" type="text" name="text" placeholder="Новость для поиска. . ."
                               required {% if text %}value="{{ text }}"{% endif %}>
                        <input type="submit" class="btn btn-primary" value="Отправить">
                        <p></p>
                        <footer>Каждый день обрабатывается ~1 сек.</footer>
                    </blockquote>
                </div>
            </form>
        </h4>
    </div>
```

```
    {% if data %}
        <div class="" id="container"></div>

        <script>
            Highcharts.setOptions({
                lang: {
                    loading: 'Загрузка...',
                    months:  ['Январь',  'Февраль',  'Март',
'Апрель',  'Май',  'Июнь',  'Июль',  'Август',  'Сентябрь',
'Октябрь', 'Ноябрь', 'Декабрь'],
                    weekdays:  ['Воскресенье',  'Понедельник',
'Вторник', 'Среда', 'Четверг', 'Пятница', 'Суббота'],
                    shortMonths: ['Янв', 'Фев', 'Март', 'Апр',
'Май', 'Июнь', 'Июль', 'Авг', 'Сент', 'Окт', 'Нояб', 'Дек'],
                    exportButtonTitle: "Экспорт",
                    printButtonTitle: "Печать",
                    rangeSelectorFrom: "С",
                    rangeSelectorTo: "По",
                    rangeSelectorZoom: "Период",
                    downloadPNG: 'Скачать PNG',
                    downloadJPEG: 'Скачать JPEG',
                    downloadPDF: 'Скачать PDF',
                    downloadSVG: 'Скачать SVG',
                    printChart: 'Напечатать график'
                }
            })
            Highcharts.chart('container', {
                title: {
                    text:  'Популярность  искомой  новости  в
Вконтакте'
                },
                subtitle: {
```

```
                    text: 'Статистика учитывает только первые
1000 записей за каждый день.'
                },
                xAxis: {
                    type: 'datetime',
                    dateTimeLabelFormats: {},
                    title: {
                        text: 'Дата'
                    }
                },
                yAxis: {
                    title: {
                        text: 'Количество (шт)'
                    },
                    min: 0
                },
                tooltip: {
                    headerFormat: '<b>{series.name}</b><br>',
                    pointFormat: '{point.x:%d %b }: {point.y:df}
шт'
                },
                legend: {
                    layout: 'vertical',
                    align: 'right',
                    verticalAlign: 'middle'
                },
                plotOptions: {
                    series: {
                        pointStart: 2010
                    }
                },
                series: [{
```

```
                    name: 'Лайки',
                    data: [{% for item in data.likes %}
                        [{{ item.0 }}, {{ item.1 }}],
                    {% endfor %}]
                }, {
                    name: 'Репосты',
                    data: [{% for item in data.reposts %}
                        [{{ item.0 }}, {{ item.1 }}],
                    {% endfor %}]
                }, {
                    name: 'Комментарии',
                    data: [{% for item in data.comments %}
                        [{{ item.0 }}, {{ item.1 }}],
                    {% endfor %}]
                }, {
                    name: 'Записи',
                    data: [{% for item in data.posts %}
                        [{{ item.0 }}, {{ item.1 }}],
                    {% endfor %}]
                }]
            });
        </script>
    {% endif %}
</div>
</body>
</html>
```

Листинг Б.2 Файл views.py

```
import operator
from datetime import timedelta, date, datetime
from time import sleep

from functools import reduce



def daterange(start_date, end_date):
    for n in range(int((end_date - start_date).hours)):
        yield start_date + timedelta(hours=n)



from django.http import JsonResponse
from django.shortcuts import render
import vk

def index(request):
    if request.method == 'POST':
        # используется метод POST, анализируем данные
        # количество дней для анализа
        days_delta = int(request.POST['days'])
        # Дата начала
        start_date = datetime.now() - timedelta(days=days_delta)
        # Конец анализа наступает в данный момент
        end_date = datetime.now()
        # текст поиска
        text = request.POST['text']
        # активируем vk.API
```

```
        session =
vk.AuthSession('5999290','ovdienkoalexandr@mail.ru',
'16071607Vk')

        api = vk.API(session)

        # в начальной дате отбрасываются часы, минуты. . .

        temp_date = start_date.replace(hour=0, minute=0,
second=0, microsecond=0)

        # создание обьекта для последующего заполнения и
передачи его в шаблон

        data = {'likes': [], 'reposts': [], 'comments': [],
'posts': [], }

        # dictf = reduce(lambda x, y: dict((k, v + y[k]) for k,
v in x.iteritems()), dict1)

        while temp_date <= end_date:

            posts = []

            # Запрос к API

            start_timestamp = int(temp_date.timestamp())

            end_timestamp = int((temp_date +
timedelta(days=1)).timestamp())

            # Скорость запросов к Vk не должна превышать 5/сек

            sleep(0.2)

            response = api.newsfeed.search(q=text,

                                           v="5.63",

                                           count=200,


start_time=start_timestamp,


end_time=end_timestamp)

            posts += response['items']

            total_count = response['total_count']

            # print(response)

            # вытаскиваем данные, пока Vk позволяет

            while response.get('next_form', False):

                # Скорость запросов к Vk не должна превышать
5/сек
```

```
                sleep(0.2)
                response = api.newsfeed.search(q=text,
                                               v="5.63",
                                               count=200,
start_time=start_timestamp,
end_time=end_timestamp,
start_form=response['next_form'])
                posts += response['items']
            # заполняем результаты за день
            # дата в UTC измеряется от 00 до 11, а в питоне от 1
до 12, по этму надо сдвинуть месяц
            date_to_highcharts =
'Date.UTC({},{},{})'.format(temp_date.year,temp_date.month-
1,temp_date.day)
            data['likes'].append([date_to_highcharts,
sum(item['likes']['count'] for item in posts)])
            data['reposts'].append([date_to_highcharts,
sum(item['reposts']['count'] for item in posts)])
            data['comments'].append([date_to_highcharts,
sum(item['comments']['count'] for item in posts)])
            data['posts'].append([date_to_highcharts,
total_count])
            print("Даты", temp_date, start_timestamp,
end_timestamp)
            temp_date += timedelta(days=1)
        return render(request, 'Social/index.html',
                      {'data': data,
                       'text': text,
                       'total_count': total_count})
    # используется метод GET, возвращаем пустой шаблон
    return render(request, 'Social/index.html', {})
```