Федеральное государственное образовательное бюджетное учреждение высшего образования

«Финансовый университет при Правительстве Российской Федерации»

**Колледж информатики и программирования**

ДОПУСКАЮ К ЗАЩИТЕ

Заместитель директора колледжа

по учебно-производственной работе

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ О.А. Потапова

« » июня 2018 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

На тему: Разработка автоматизированной системы по сбору и обработке данных в социальной сети

Студент группы 4ПКС-214

|  |  |
| --- | --- |
| Шимич Федор Алексеевич | \_\_\_\_\_\_\_\_\_ «\_\_\_» июня 2018 г. |

Основная профессиональная образовательная программа по специальности

09.02.03 Программирование в компьютерных системах

Форма обучения очная

Руководитель ВКР \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Морозова М.В.

Председатель предметно-цикловой комиссии \_\_\_\_\_\_\_\_\_\_\_\_\_Пестов А.И.

Москва

2018

Костыль для двусторонней печати задания ВКР

Содержание

[Введение 4](#_Toc516662824)

[Глава 1 Теоретическая часть 6](#_Toc516662825)

[1.1 Предпроектное исследование предметной области 6](#_Toc516662826)

[1.2 Платформы 7](#_Toc516662827)

[1.3 Требования к программному обеспечению 9](#_Toc516662828)

[1.4 Характеристика системы программирования 10](#_Toc516662829)

[1.5 Системные требования 16](#_Toc516662830)

[1.6 Назначение и основные возможности приложения 16](#_Toc516662831)

[Глава 2 Практическая часть 22](#_Toc516662832)

[2.1 Анализ требований и определение спецификаций ПО 22](#_Toc516662833)

[2.2 Проектирование вспомогательной части 25](#_Toc516662834)

[2.3 Проектирование базы данных 27](#_Toc516662835)

[2.4 Проектирование серверной части 29](#_Toc516662836)

[2.5 Разработка пользовательского интерфейса приложения 31](#_Toc516662837)

[2.6 Тестирование приложения 38](#_Toc516662838)

[2.7 Руководство системного программиста 40](#_Toc516662839)

[2.8 Руководство программиста 41](#_Toc516662840)

[2.9 Руководство пользователя 42](#_Toc516662841)

[Заключение 44](#_Toc516662842)

[Список используемой литературы 45](#_Toc516662843)

[Приложение А 47](#_Toc516662844)

[Приложение Б 48](#_Toc516662845)

[Приложение В 49](#_Toc516662846)

# Введение

Сегодня социальные сети стали неотъемлемой частью жизни многих людей. В России проникновение соцсетей оценивается в 47%, аккаунты в них имеют 67,8 млн россиян. По данным Statista, активнее всего в РФ используют YouTube (63% опрошенных), второе место занимает ВКонтакте — 61%. Количество пользователей с каждым днем растут. В социальных сетях контент (контент – это информационное содержание сайта (тексты, графическая, звуковая информация и др.), а также книги, газеты, сборники статей, материалов и др.) придумывают, создают и выкладывают пользователи, коммерческие и некоммерческие организации.

Публичные страницы в социальной сети Вконтакте широко распространены и активно используются пользователями, но для данной социальной сети характерен низкий отклик пользователей на информацию, которую им предоставляет администраторы публичных страниц, с чем это связано заслуживает отдельной исследовательской работы, но факт остается фактом. Именно поэтому необходимо очень кропотливо подходить к анализу статистики взаимодействия пользователя с публичной страницей. Вконтакте предоставляет достаточно неплохую нативную статистику, но иногда встроенной функциональности не хватает и необходимо обращаться к сторонним сервисам. Среди сторонних сервисов можно выделить две категории – приложения, рассчитанные исключительно на профессионалов в сфере социального маркетинга и узкоспециализированные приложения, решающие только одну задачу. Именно поэтому было принято решение создать простое, но в тоже время информативное приложение, которое помогло бы администраторам публичных страниц без особых усилий понять вектор их направления, в котором стоит развивать сообщество. Главное это понятный интерфейс и прозрачная логика работы, чтобы приложением могли пользоваться как профессионалы, так и простые пользователи, например, художник, который публикует свои работы.

Основная цель проекта – это помочь пользователям понять правильно ли они развивают свой ресурс.

Основные задачи:

* Продумать структуру программного комплекса;
* Выбрать платформу и средства реализации проекта;
* Подобрать модули с соответствующей лицензией (MIT);
* Подготовить сервер и настроить его;
* Разработать структуру базы данных;
* Настроить сервер с базой данных;
* Разработать автоматизированную часть приложения;
* Разработать приложение клиент-сервер;
* Разработать пользовательский интерфейс;

Основным источником информации будет техническая документация, которая поставляется с программными модулями и техническая литература.

# Теоретическая часть

* 1. Предпроектное исследование предметной области
     1. Социальная сеть

Социальная сеть — веб-приложение, созданное для построения сообщества пользователей, сети и организации удобного взаимодействия между ними. Контент социальной сети создается пользователями.

В России проникновение соцсетей оценивается в 47%, аккаунты в них имеют 67,8 млн россиян. По данным Statista, активнее всего в РФ используют YouTube (63% опрошенных), второе место занимает ВКонтакте — 61%. Глобальный лидер Facebook лишь на четвертой строчке с показателем в 35%. На рисунке 1.1 представлен график статистики Statista на начало 2018 года.

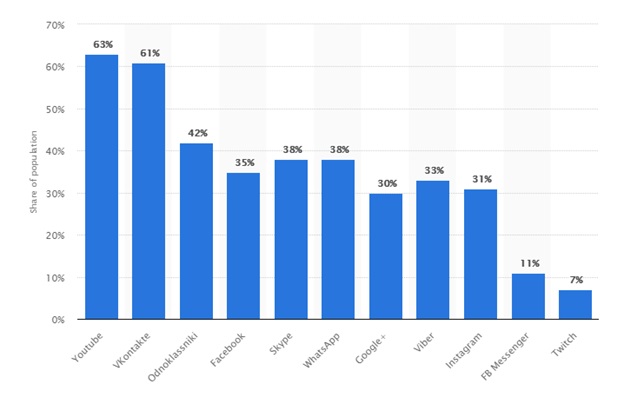


Рис. 1.1 - график статистики Statista на начало 2018 года

Объектом исследования выбрана одна из самых популярных социальных сетей в России – Вконтакте.

1.1.2 Вконтакте

Вконтакте — российская социальная сеть со штаб-квартирой в Санкт-Петербурге. Сайт доступен на более чем 90 языках; особенно популярен среди русскоязычных пользователей. ВКонтакте позволяет пользователям отправлять друг другу сообщения, создавать собственные страницы и сообщества, обмениваться изображениями, тегами, аудио- и видеозаписями, играть в браузерные игры.

* + 1. Публичные станицы и группы

Все термины применимы только в рамках экосистемы социальной сети Вконтакте.

Группы — это тематические клубы, сообщества друзей, коллег по работе и учёбе. ВКонтакте создано более 16 миллионов групп, среди которых можно искать интересующие в поиске или создать свою.

Публичные страницы являются инструментом именно публичного представления в сети, в отличие от групп, которые могут быть закрытыми. Эти страницы по умолчанию доступны для незарегистрированных или неавторизованных в сети ВКонтакте пользователей.

Страница поможет компаниям и известным персонам быть в контакте со своими клиентами или поклонниками, а им, в свою очередь, получать новости «из первых рук» внутри сети ВКонтакте.

## Платформы

Одно из главных решений разработчика – это выбрать платформу, для которой будет разработано приложение. Для этого необходимо проанализировать состояния рынка на данный момент.

На рисунке 1.2 представлена статистика операционных систем в России за период с января 2018 года по май 2018 года.

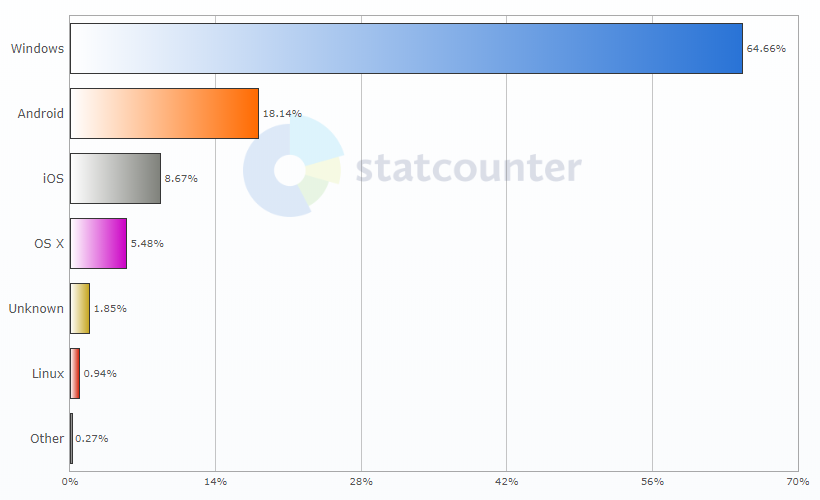


Рис. 1.2 – Статистика ОС statcounter

Поскольку большинство пользователей используют разные операционные системы, необходимо написать универсальное решение, чтобы пользователь любой операционной системы имел возможность полноценно пользоваться приложением. Одно из решений – это веб-приложение.

Далее приведены преимущества веб-приложения:

* Пользователь не нуждается в установке программного обеспечения. Все, что требуется для полноценной работы - это браузер и доступ в Интернет. Устанавливая приложения на свой компьютер, поневоле приходится брать на себя обязанности администратора, что доставляет неопытным пользователям массу проблем. Приложение нужно установить и запустить, потом настроить под себя, а потом ни с того ни с чего вдруг возникают непонятные ошибки, требующие немедленного решения. В случае с веб-приложением такие проблемы не возникают.
* Веб-приложения не требовательны к ресурсам и не предъявляют никаких требований к аппаратной платформе.
* Нет никаких проблем с поддержкой старых версий программ и обратной совместимостью. Когда появляется новая версия десктопного приложения, пользователям нередко приходится решать проблемы, связанные с обновлением уже установленной на их машине копии. В случае с браузерными приложениями таких проблем не возникает - существует только одна версия, в которой работают все пользователи, и в случае выхода новой все без исключения автоматически переходят на нее, порой даже не замечая этого.

На рисунке 1.3 представлена статистика браузеров, используемых в России, исходя из нее видно, что более 95% пользователей используют современные браузеры, для которых можно написать полноценное веб-приложение.

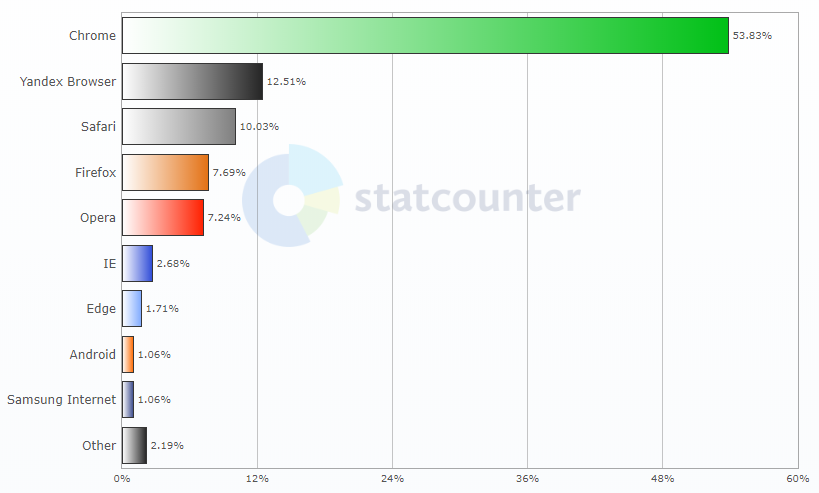


Рис. 1.3 – Статистика браузеров statcounter

* 1. Требования к программному обеспечению

Функциональные требования к приложению:

* Приложение должно работать во всех современных браузерах;
* Получение секретного ключа пользователя (токен) в автоматическом режиме;
* Визуализация данных с помощью графиков;
* Таблицы с возможностью сортировки;
* Галерея для просмотра изображений;
* Возможность задать выборку из предложенных данных (фильтр);
* Добавление и удаление отслеживаемых групп;
* Предоставлять доступ к API только авторизированным пользователям;
* Реализовать основной функционал в рамках одной страницы с использованием технологии AJAX;
* Возможность выбора группы по которой будут выводится информация.

Нефункциональные требования к приложению:

* Стабильная работа;
* Быстрая загрузка страниц;
* Простой и понятный интерфейс;
* Добавить справку к основному функционалу приложения;
  1. Характеристика системы программирования

1.4.1 Web-приложение

Веб-приложение — клиент-серверное приложение, в котором клиент взаимодействует с сервером при помощи браузера, а за сервер отвечает — веб-сервер.

Веб-приложение представляет собой веб-сайт, на котором размещены страницы с частично либо полностью несформированным содержимым. Окончательное содержимое формируется только после того, как посетитель сайта запросит страницу с веб-сервера. В связи с тем, что окончательное содержимое страницы зависит от запроса, созданного на основе действий посетителя, такая страница называется динамической.

1.4.2 MVC

Шаблон архитектуры Model-View-Controller (MVC) разделяет приложение на три основных компонента: модель, представление и контроллер (Рис. 1.4).

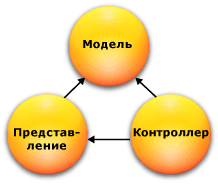


Рис. 1.4 - MVC

В состав платформы MVC входят следующие компоненты:

* Модели. Объекты моделей являются частями приложения, реализующими логику для домена данных приложения. Объекты моделей часто получают и сохраняют состояние модели в базе данных. В небольших приложениях эта модель подразумевает концептуальное, а не физическое разделение. Например, если приложение только считывает набор данных и отправляет его в представление, то физический слой модели и связанных классов отсутствует. В этом случае набор данных принимает роль объекта модели.
* Представления. Представления служат для отображения пользовательского интерфейса приложения. Пользовательский интерфейс обычно создается на основе данных модели.
* Контроллеры. Контроллеры осуществляют взаимодействие с пользователем, работу с моделью, а также выбор представления, отображающего пользовательский интерфейс. В приложении MVC представления только отображают данные, а контроллер обрабатывает вводимые данные и отвечает на действия пользователя. Например, контроллер может обрабатывать строковые значения запроса и передавать их в модель, которая может использовать эти значения для отправки запроса в базу данных.

1.4.3 Веб-фреймворк

Современным способом разработки интернет ресурсов является применение MVC фреймворков. Фреймворк является архитектурой разрабатываемого ресурса, который гарантирует стандартную структуру программ и их поведение по умолчанию. Данные в MVC отделены от реализации функционала, а тот, в свою очередь, – от представления (внешнего вида).

В общем случаи веб-фреймворк состоит из следующих компонентов:

* Шаблонизатор. Отвечает за независимость вёрстки от программного кода.
* Роутер. Распознаёт URL, по которому произошло обращение к серверу.
* Модуль доступа к базе данных.
* Модуль кэширования. Ускоряет загрузку страниц.
* Модуль безопасности. Аутентификация и авторизация пользователей.
* Файлы конфигурации. Фреймворки также могут управлять сессиями,
* Вести логирование, упрощать использование технологии
* Ajax и многое другое.

Таким образом, делая сайт на базе фреймворка, мы получаем доступ к большому количеству готовых функций. Все базовые функции – от доступа к базе данных до отдачи готовой страницы браузеру – уже написаны создателями фреймворка. Среди популярных современных веб-фреймворков можно отметить следующие: Zend Framework, Yii, Symfony2, Laravel, CakePHP (PHP), Django (Python), Ruby on Rails (Ruby).

Для своего приложения был сделан выбор в сторону Django. Сравнив его с другими решениями могу выделить достоинства:

* Использование Python в качестве языка программирования. Поддержка метапрограммирования, обширная библиотека классов, хорошая документация и достаточно компактный и интуитивно понятный синтаксис.
* Великолепная документация — множество примеров, объяснений, и самое главное — открытый исходный код, который очень хорошо написан.
* Встроенный ORM (Object-relational mapper). В абсолютном большинстве случаев совершенно не требуется использование SQL-синтаксиса в выражениях, что автоматом снижает риск появления SQL-injection уязвимости.
* Автоматически генерируемая панель администратора. Это одна из уникальных возможностей Django, у которой практически нет аналогов. Помимо того, что данная функциональность позволяет значительно сократить время на написание нужного интерфейса, она еще и дает возможность клиентам сразу начать работать с сайтом еще на начальных этапах его разработки! Фактически, достаточно набросать нужные модели, и можно сразу показывать сайт клиенту, и уже интерактивно с ним обсуждать бизнес-логику, не отвлекаясь на дизайн.
* Поддержка MTV (Model-Template-View). Данный паттерн проектирования очень близок к классическому MVC, и самое главное, что он позволяет — это хорошо отделять бизнес-логику от дизайна.
* Высокая скорость работы. Несмотря на то, что Python не блещет скоростью, в целом Django работает достаточно быстро. Он может выдерживать высокую нагрузку, плюс имеет встроенные возможности кэширования и распределения нагрузки.

1.4.4 Веб-фреймфорк VueJS

VueJS — это прогрессивный фреймворк для создания пользовательских интерфейсов. В отличие от фреймворков-монолитов, Vue создан пригодным для постепенного внедрения. Его ядро в первую очередь решает задачи уровня представления (view), что упрощает интеграцию с другими библиотеками и существующими проектами. С другой стороны, Vue полностью подходит и для создания сложных одностраничных приложений (SPA, Single-Page Applications), если использовать его совместно с современными инструментами и дополнительными библиотеками.

1.4.5 Apache

Apache2 - это HTTP сервер, обладающий высокой надежностью и гибкостью, под HTTP сервером следует понимать программное обеспечение для обработки HTTP запросов. Основная работа Apache это обработка и ответ на HTTP запросы и генерирование динамическое содержание страниц. Гибкость достигается путем использования файла .htaccess, благодаря которому можно переопределять глобальные настройки сервера Apache.

Mod\_wsgi – это модуль для веб-сервера Apache, который предоставляет WSGI-совместимый интерфейс для работы с web-приложениями, написанными на языке программирования Python. Необходим для работы с Django.

1.4.6 Django

Django - свободный фреймворк для веб-приложений, написанных на Python, который реализован на основе архитектуры MVC.

Отличительные особенности джанго:

* любой запрос обрабатывается в программе и перенаправляется на свой адрес (url);
* разделение контента и представления с помощью шаблонов;
* абстрагирование;
* от низкого уровня баз данных.

Джанго-приложение состоит из четырех основных компонентов.

Модель данных: данные являются сердцевиной любого современного Web-приложения. Модель – важнейшая часть приложения, которое постоянно обращается к данным при любом запросе из любой сессии. Объектно-ориентированный маппер (ORM) обеспечивает таким классам доступ непосредственно к базам данных. Если бы не было ORM, программисту пришлось бы писать запросы непосредственно на SQL. Модель обеспечивает облегченный механизм доступа к слою данных, инкапсулирует бизнес-логику. Модель не зависит от конкретного приложения. Данными можно манипулировать даже из командной строки, не используя при этом Веб-сервер.

Представление (view): вьюхи в джанге 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.

Шаблоны: они 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 регулярные выражения делают механизм достаточно гибким. При этом одно представление может быть сконфигурировано к нескольким url, предоставляя доступ различным приложениям. Здесь поддерживается философия закладок: url становятся как бы самодостаточными и начинают жить независимо от представления.

1.4.7 Webpack

Webpack — это сборщик модулей для современных JavaScript-приложений. Когда вебпак обрабатывает Ваше приложение, он рекурсивно создает граф зависимостей, включающий все модули, необходимые приложению, затем упаковывает все эти модули в заданное число файлов — часто всего в один — для загрузки браузером.

Webpack берет модули с зависимостями и генерирует статические ресурсы, которые представляют эти модули.

## Системные требования

Минимальные системные требования к серверной части:

* Процессор: Intel(R) Xeon CPU E5-2690 2.90 GHz (4 ядра);
* Платформа: 32-х или 64-х разрядная;
* Оперативная память: Не менее 5 Gb;
* Жесткий диск: Не менее 2000 Mb свободного объема.

Поддерживаются следующие версии ОС семейства Linux:

* Debian;
* Ubuntu;
* Red Hat Enterprise Linux;
* Astra Linux.

Минимальные системные требования к клиентской части:

* Веб-браузер с поддержкой JavaScript;
* Оперативная память: Не менее 512 Mb.

## Назначение и основные возможности приложения

Публичные страницы (Рис. 1.5) в социальной сети Вконтакте широко распространены и активно используются пользователями.

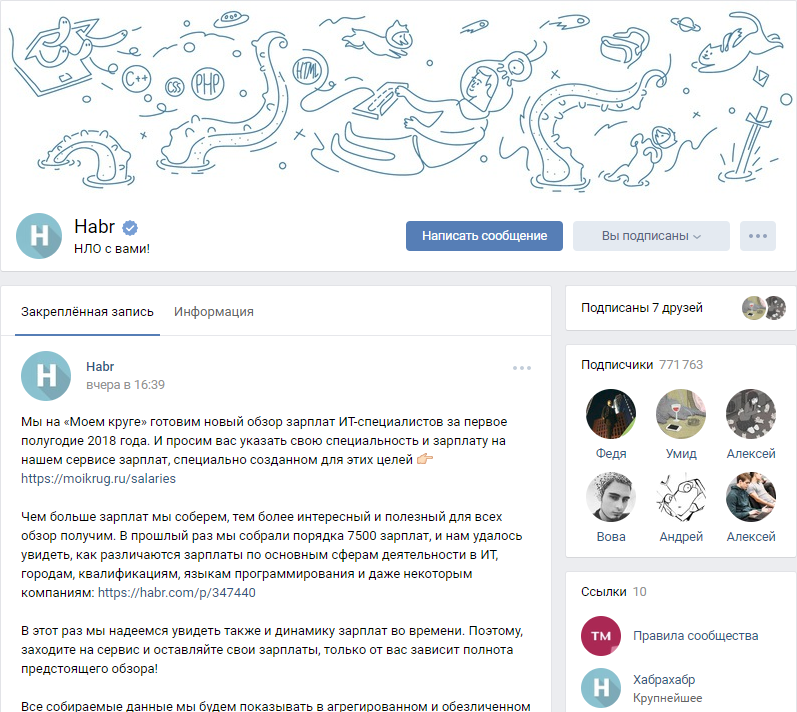


Рис. 1.5 – пример сообщества

Для данной социальной сети характерен низкий отклик пользователей на информацию, которую им предоставляет администраторы публичных страниц, с чем это связано заслуживает отдельной исследовательской работы, но факт остается фактом. Именно поэтому необходимо очень кропотливо подходит к анализу статистики взаимодействия пользователя с публичной страницей. Вконтакте предоставляет достаточно неплохую нативную статистику (Рис. 1.6), но иногда встроенного функционала не хватает и необходимо обращаться к сторонним сервисам.



Рис. 1.6 – пример статистики Вконтакте

Среди сторонних сервисов можно выделить две категории – приложения, рассчитанные исключительно на профессионалов в сфере социального маркетинга и узкоспециализированные приложения, решающие узкий спектр задач. Именно поэтому было принято решение создать простое, но в тоже время информативное приложение, которое помогло бы администраторам публичных страниц без особых усилий понять вектор их направления, в котором стоит развивать сообщество. Главное это понятный интерфейс и прозрачная логика работы, чтобы приложением могли пользоваться как профессионалы, так и простые пользователи, например, художник, который публикует свои работы.

Одна из главных составляющих успешного сообщества – это подписчики, которые следят за новостями и взаимодействуют с ним. Для этого необходим механизм отслеживания прироста и оттока подписчиков. В Вконтакте он есть, но он ограничен выводом количества вступивших и вышедших – можно увидеть на рисунке 1.7.

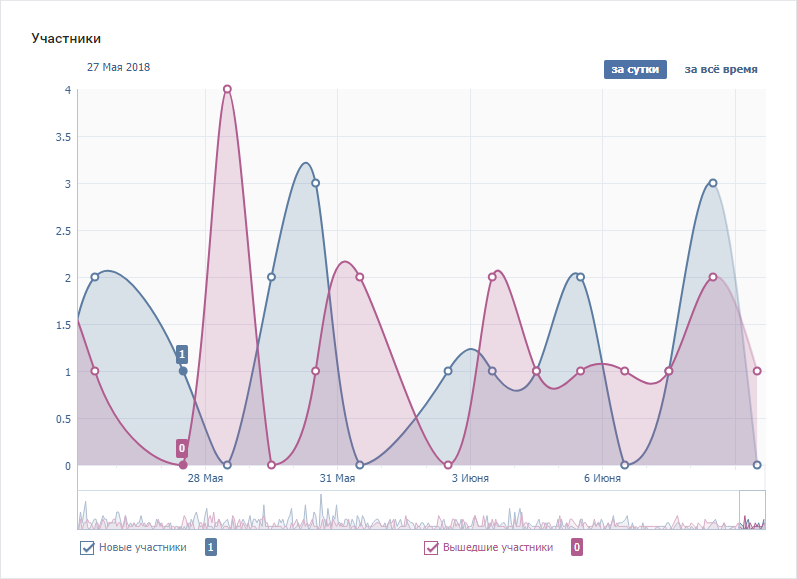


Рис. 1.7 – пример статистики участников

У нас же эта информация более подробная и предоставляет возможность просматривать кто конкретно вступил и вышел, а также кто вернулся в сообщество (Рис. 1.8). Например, такие данные могут помочь при анализе успешности рекламной компании.

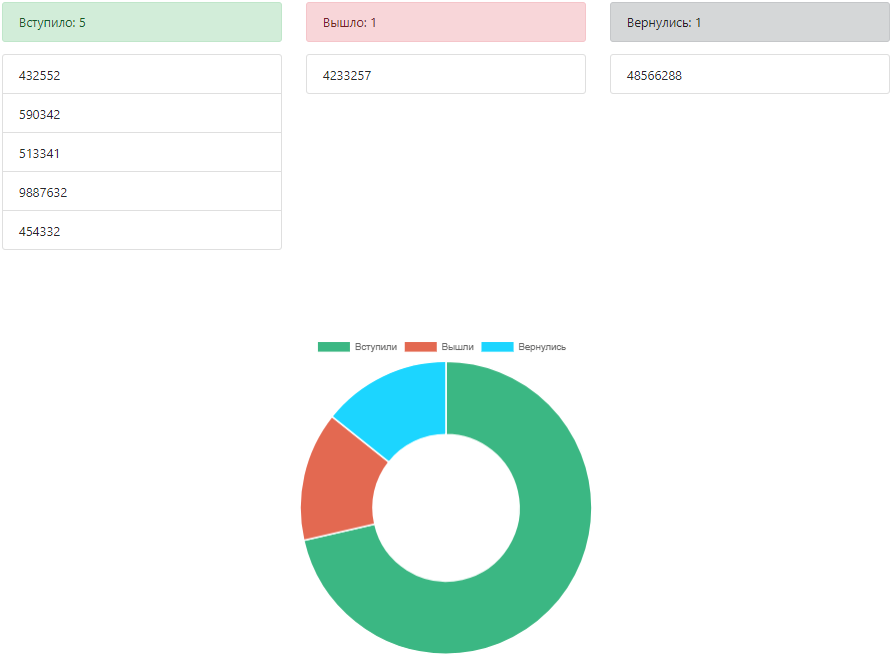


Рис. 1.8 – Статистика участников в приложение

Другая не менее важная составляющая хорошего сообщества – это контент. Большая его часть публикуется на стену в виде постов (Рис. 1.9), которые в дальнейшем попадают пользователю в новостную ленту.

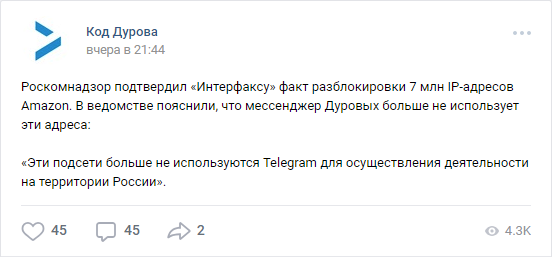


Рис. 1.9 – пример поста Вконтакте

Поскольку социальная сеть понимает важность этой составляющей, она разработала подробную статистику постов. Поэтому для упрощения анализа администраторам приложения собирает обобщённую статистику (Рис. 1.10) за разные периоды времени – это поможет сэкономить время.

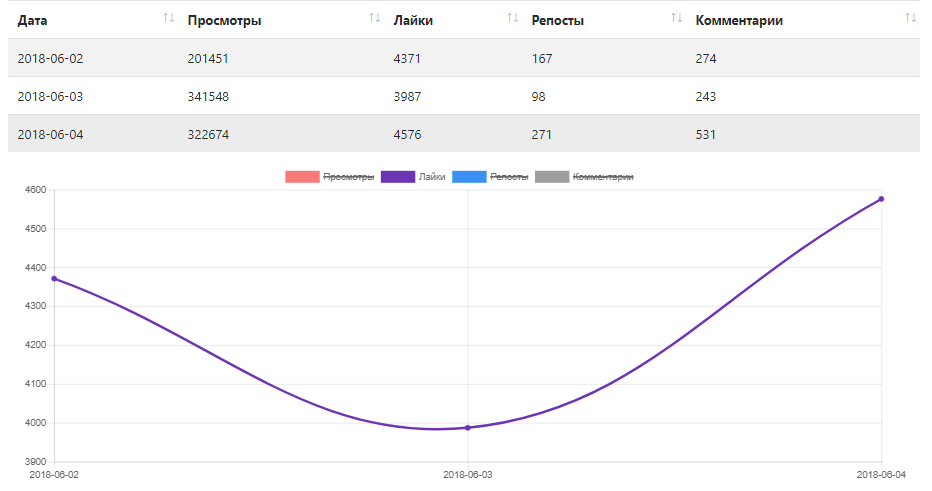


Рис. 1.10 – статистика постов в приложение

Так же всегда хочется узнать какие посты собирают большую активность вокруг себя, но для этого необходимо затратить немного времени, но все же это время. Именно поэтому было решено упростить этот процесс до пары кликов мышки. Теперь достаточно выбрать сообщество, задать фильтр (Рис. 1.11) и получить посты которые выше среднего или ниже.



Рис. 1.11 – фильтр в приложение

# Практическая часть

## Анализ требований и определение спецификаций ПО

Приложение можно разделить на 4 части - база данных, вспомогательная часть, серверная часть и клиентская часть.

Требования, предъявляемые к базе данных:

* Хранение большого количества информации;
* Быстрая обработка запросов на запись и получение информации;
* Отказоустойчивость.

Вспомогательная часть – это процесс, работающий в фоновом режиме и взаимодействующий только с базой данных.

Требования, предъявляемые к вспомогательной части:

* Работа в фоновом режиме;
* Работа с базой данных.

Серверная часть – это обработка запросов, поступающих из сети интернет по протоколу TCP/IP, ввод и вывод информации из базы данных, предоставления статических файлов для пользовательской части.

Требования, предъявляемые к серверной части:

* Обрабатывать запросы через TCP/IP;
* Читать и записывать информацию из базы данных;
* Отправлять пользователю готовую html страницу;
* Работать в режиме RestAPI;
* Предоставлять доступ к статическим данным.

Пользовательская часть. Обработка и визуализация данных из формата JSON в пользовательский интерфейс. Взаимодействие с серверами с помощью RestAPI.

На рисунке 2.1 представлена диаграмма развёртывания. Это рекомендуемая конфигурация компонентов и серверов для корректной и стабильной работы автоматизированной системы. При необходимости есть возможность устанавливать компоненты на один сервер – это зависит от нагрузки на проект и бюджета, выделенного на сервера.

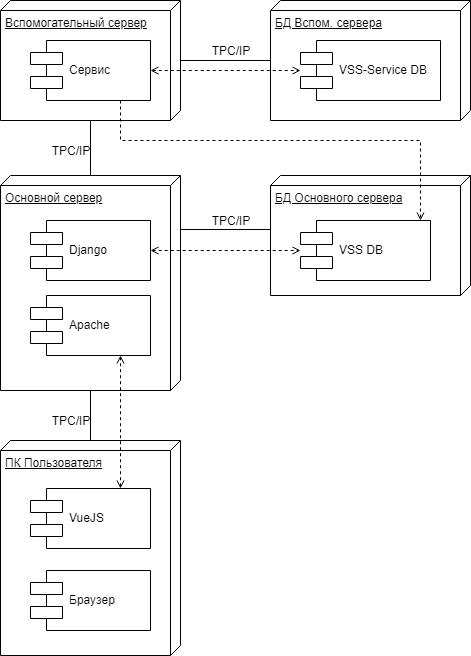


Рис. 2.1 – Диаграмма развёртывания

На рисунке 2.2 представлена контекстная диаграмма потоков данных. На ней отражен только поток данных, который связан с пользователем.

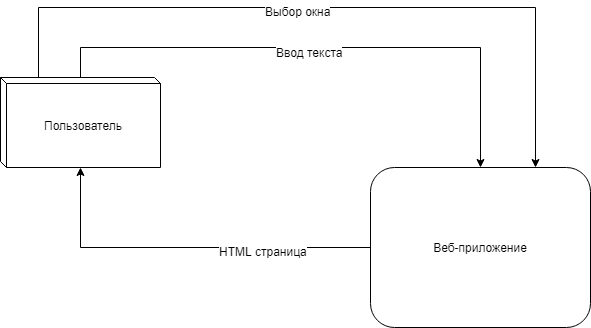


Рис. 2.2 – Диаграмма потоков данных

На рисунке 2.3 представлена контекстная функциональная диаграмма.

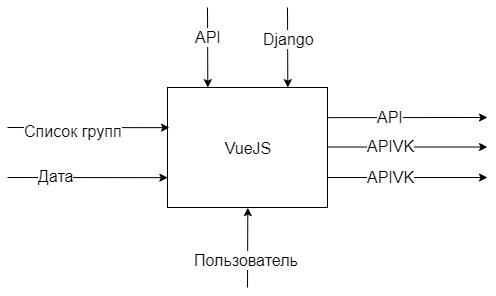


Рис. 2.3 - Контекстная функциональная диаграмма

На рисунке 2.4 представленная диаграмма вариантов использования, на которой отражено взаимодействие пользователя и приложения.

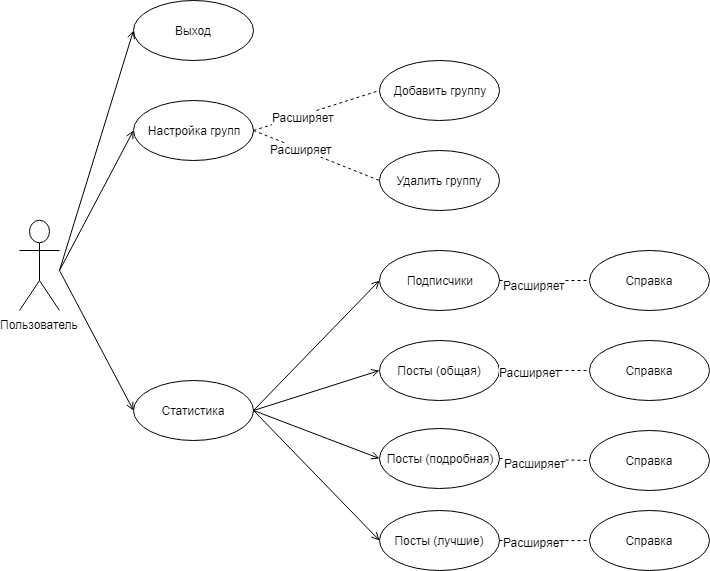


Рис. 2.4 – Диаграмма вариантов использования

## Проектирование вспомогательной части

Исходя из предъявленных требований, было принято решение написать два скрипта на языке программирования Python 3. На рисунке 2.5 представлена функциональная схема вспомогательной части приложения – сервиса.

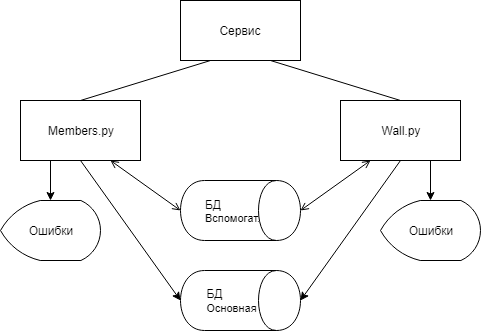


Рис. 2.5 – Функциональная схема сервиса

Жизненный цикл каждого из скрипта состоит из 4 этапов:

1. Получение списка индикаторов групп;
2. Поэтапное получение информации с помощью API VK;
3. Обработка полученной информации и запись в базу данных сервиса;
4. Кеширование полученной информации и запись в базу данных сервера.

В рамках первого этапа жизненного цикла необходимо получить список групп и секретные ключи безопасности (токены) пользователя. Ключи нужны, чтобы в дальнейшем обращаться к APIVK. В листинге 2.1 можно посмотреть на SQL запрос к базе данных который вернет список пользователей с их ключами.

Листинг 2.1 – получение списка групп

|  |
| --- |
| query = "SELECT u.vk\_token, g.group\_id FROM vss.user\_info AS u RIGHT JOIN vss.user\_info\_group\_id AS g ON u.user\_id = g.user\_id" |

Для получения информации с серверов Вконтакте необходимо делать GET-запрос по адресу «https://api.vk.com/method/» одним из обязательных параметров – токен пользователя. Ответ возвращается в формате JSON и дальше обрабатывается скриптом.

Кеширование необходимо для уменьшения нагрузки на базу данных и для отзывчивости запросов. Например, в таблице с информацией о подписчиках одновременно может храниться от 10 миллионов строк, и выборка из этой таблицы может занять более 20 секунд, что недопустимо для приложения, именно поэтому был придуман механизм кеширования и запросы выполняются в 100 раз быстрее, в зависимости от нагрузки на сервер.

## Проектирование базы данных

Исходя из требований необходимо спроектировать две базы данных под разные задачи. Для реализации задачи была выбрана СУБД MySQL – которая в полной мере удовлетворяет требованиям и является бесплатным продуктом.

На рисунке 2.6 представлена диаграмма классов для базы данных серверной части. На ней представлены основные компоненты, подробная диаграмма представлена в приложении А.

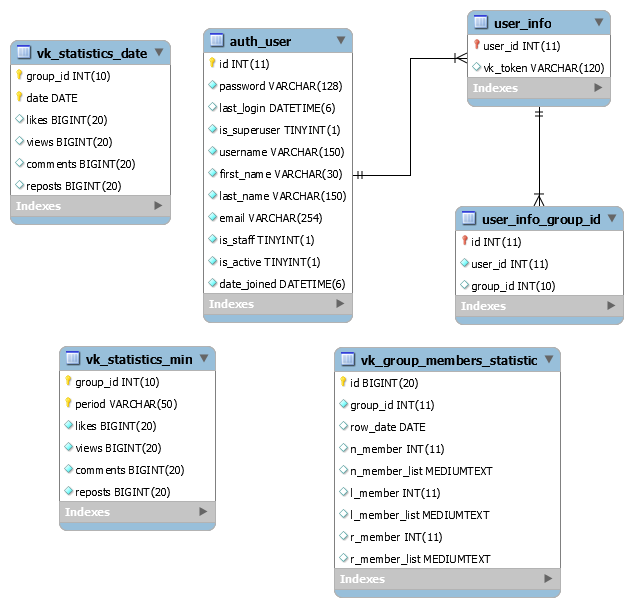


Рис.2.6 – Диаграмма классов для серверной БД

На рисунке 2.7 представлена диаграмма классов для базы данных вспомогательной части – сервиса.

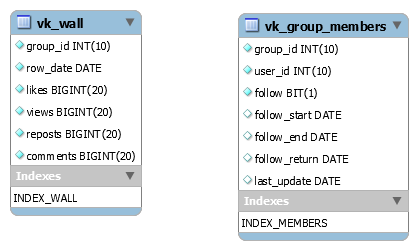


Рис. 2.7 – диаграмма классов вспомогательной БД

В нее входит всего две таблицы, которые хранят основную информацию о подписчиках и статистику постов за каждый день.

## Проектирование серверной части

Для реализации серверной части были выбраны технологии:

* Apache2;
* mod\_wsgi;
* Django;

Apache2 решает задачи:

* Обрабатывать запросы через TCP/IP;
* Отправлять пользователю готовую html страницу для отображения
* Предоставлять доступ к статическим данным

Django решает задачи:

* Читать и записывать информацию из базы данных
* Отправлять пользователю готовую html страницу для отображения
* Работать в режиме RestAPI.

На рисунке 2.8 представлена структурная схема серверной части. Она отражает совокупность элементарных звеньев объекта и связей между ними.

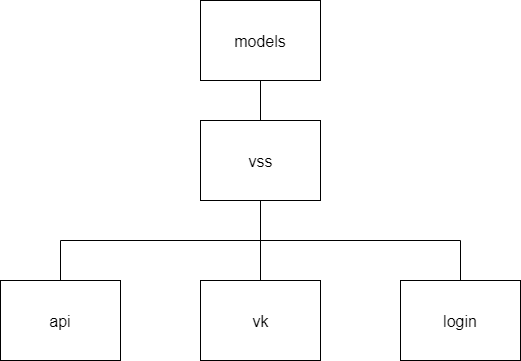


Рис. 2.8 - Структурная схема серверной части

Краткое описание компонентов:

* models – это модель представления базы данных.
* vss – это основной модуль в котором представлены все настройки и зависимости проекта.
* api – это модуль который предоставляет информацию в формате JSON и работает в рамках технологии REST API.
* vk – это модуль который реализует паттерн MVC. Генерация динамических html документов.
* auth – модуль проверки авторизации пользователя. Предотвращает неправомерный доступ к данным.

На рисунке 2.9 представлена диаграмма компонентов. Она отражает как взаимодействуют компоненты между собой.

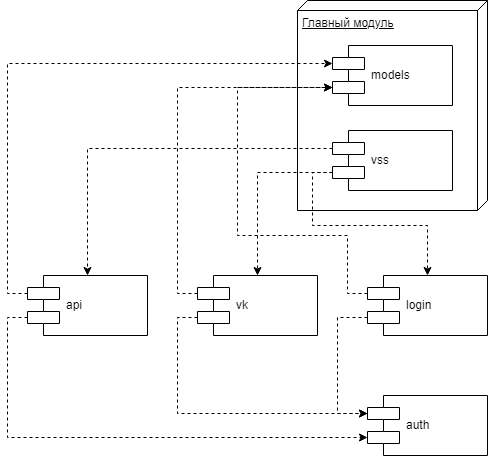


Рис. 2.9 – диаграмма компонентов

## Разработка пользовательского интерфейса приложения

У приложения имеются два основных раздела – статистика и настройка групп. При переходе по адресу на котором находится приложение открывается страница статистики. На каждой странице обязательно присутствует шапка приложения ее можно увидеть на рисунке 2.10. В разделе статистики пользователь может выбрать отслеживаемую группу и тип статистики, в соответствие с выбранными пунктами отобразится информация.

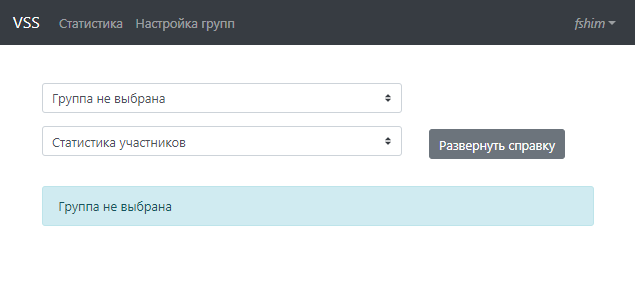


Рис. 2.10 – Шапка приложения

Если у пользователя не настроены отслеживаемые группы, то приложение выводит соответствующие сообщение (Рис. 2.11), в котором пользователю будет предложено перейти в раздел настройки групп.

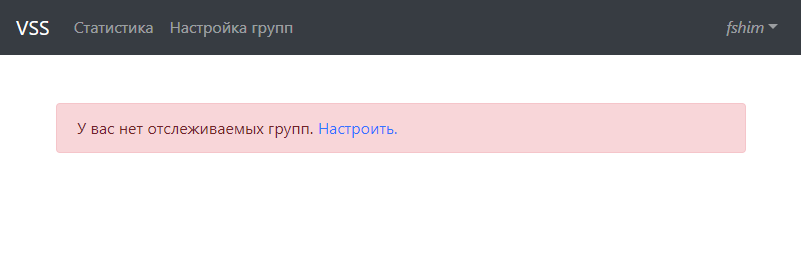


Рис. 2.11 – Сообщение

Для каждого раздела на странице статистики есть справка (Рис. 2.12) в которой находится краткая информация о разделе.

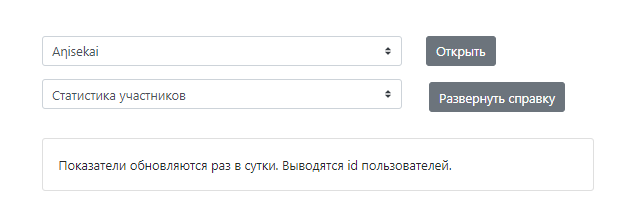


Рис. 2.12 - Справка

Статистика участников - раздел, включенный в страницу статистики (Рис. 2.13), здесь пользователю предлагается выбрать временной период, за который необходимо отобразить информацию. Так же выводится информация о подписчиках выбранной группы и график, по которому можно понять соотношение вышедших, вступивших и вернувшихся подписчиков группы.

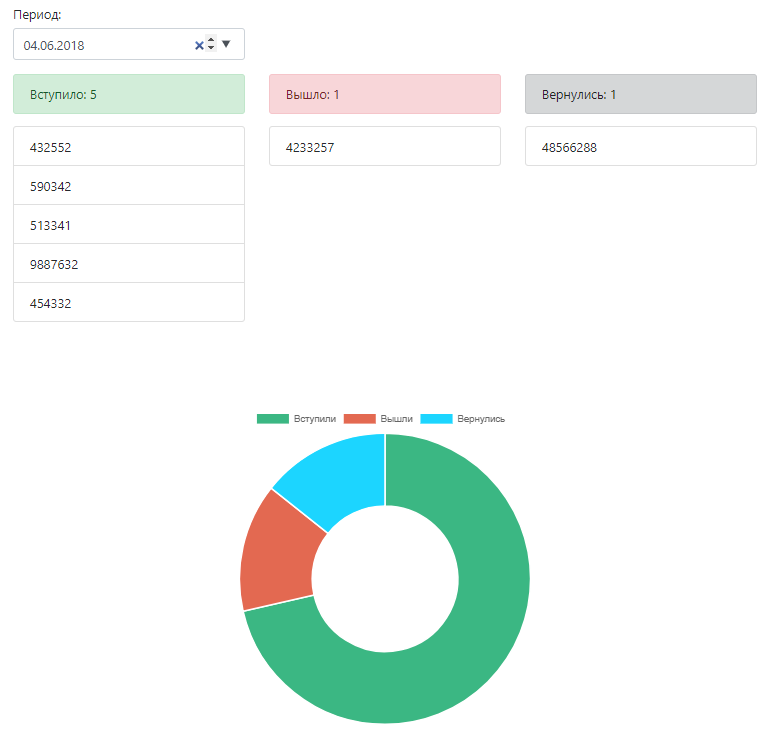


Рис. 2.13 – Статистика участников

Статистика контента – раздел, включенный в страницу статистики (Рис. 2.14), здесь пользователю выводится обобщенная информация по средним показателям просмотров, лайков, репостов и комментариев за разные периоды времени.

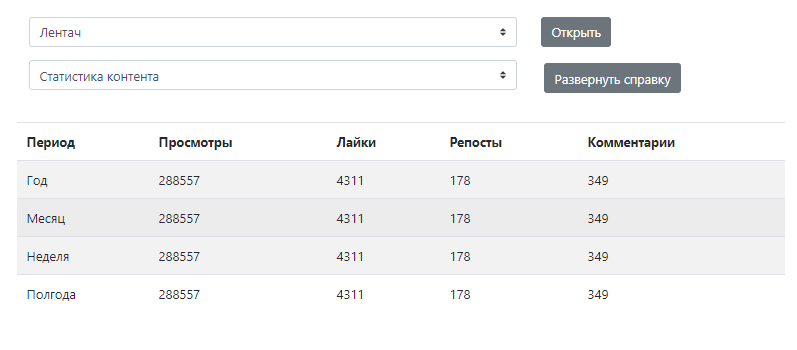


Рис. 2.14 – Статистика контента

Подробная статистика контента – раздел, включенный в страницу статистики (Рис. 2.15), здесь пользователю выводится информация по средним показателям просмотров, лайков, репостов и комментариев за разные периоды времени, есть возможность сортировки в таблице по одному показателей. Так же предлагается интерактивный график, который визуализирует данные.

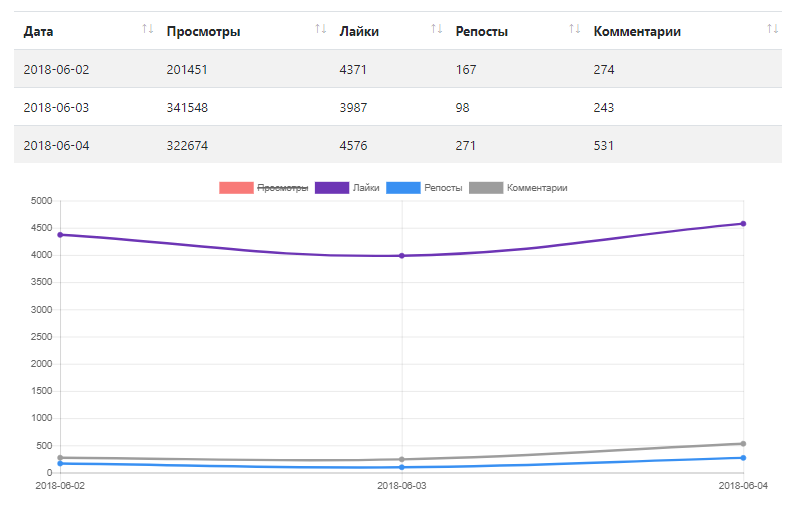


Рис. 2.15 - Подробная статистика контента

Подробная статистика контента – раздел, включенный в страницу статистики (Рис. 2.16), здесь пользователю выводится последние 100 постов с подробной информацией из выбранной группы.

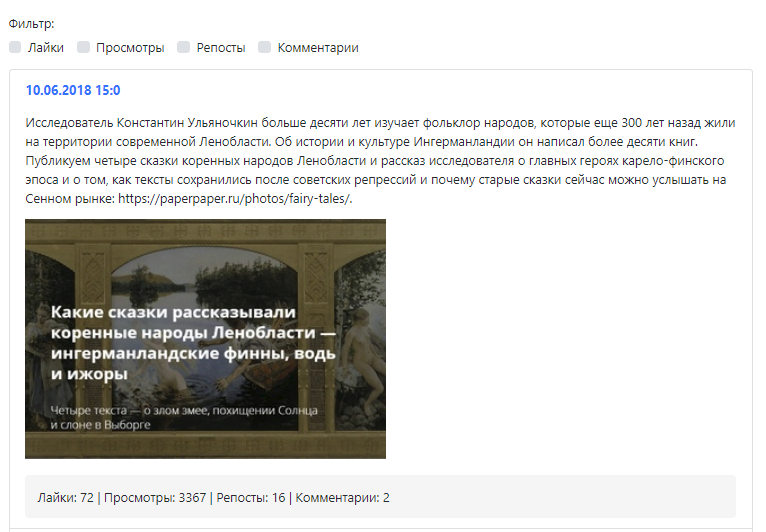
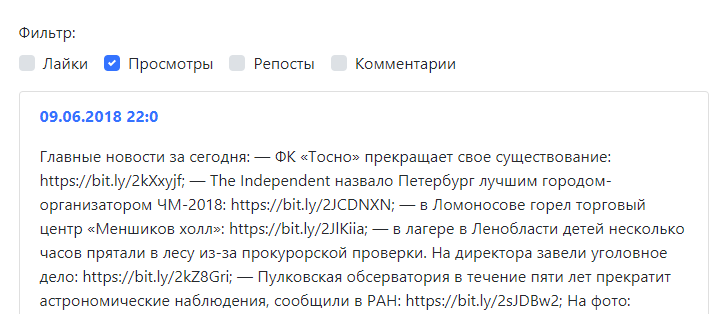


Рис. 2.16 - Подробная статистика контента

Суть этого раздела в том, что ползователь может выбрать фильтры, по которым будут выводится записи (Рис. 2.17). Например, если пользователю необходимо узнать последние самые просматривемые записи, то он задает фильтер «Просмотры», и в режиме реального времени ему выведутся самые просматриваемые посты.

Рис. 2.17 - Фильтр

Так же в этом разделе есть возможность просмотра иллюстраций постов с помощью галереи (Рис. 2.18)

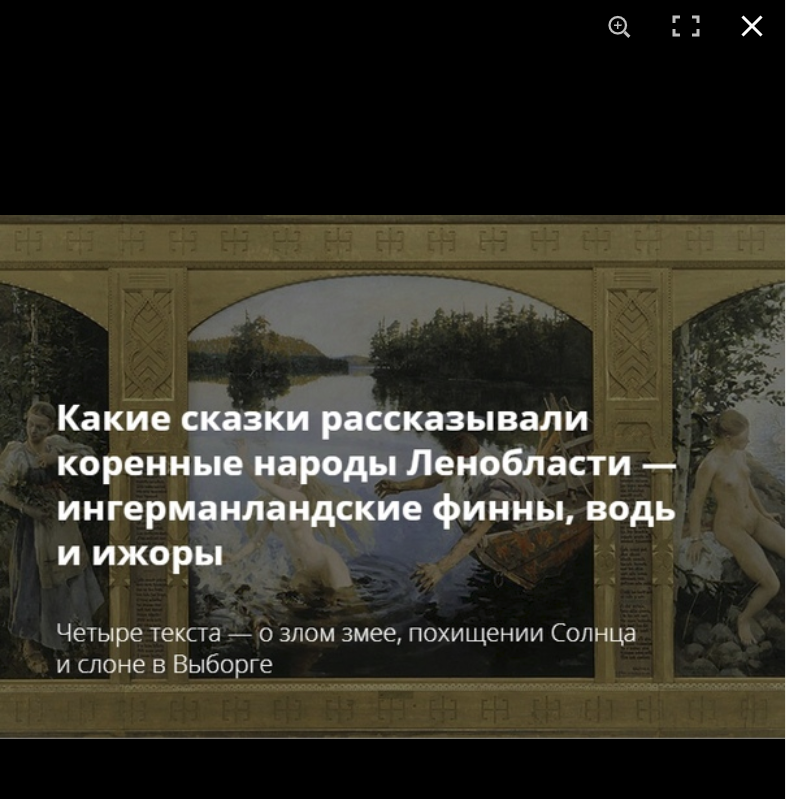


Рис. 2.19 – Галерея

На странице отслеживаемые группы (Рис. 2.20) пользователь может настроить группы, по которым в дальнейшем будет собираться статистика в автоматическом режиме, так же они станут доступны для выбора на страницы статистики.

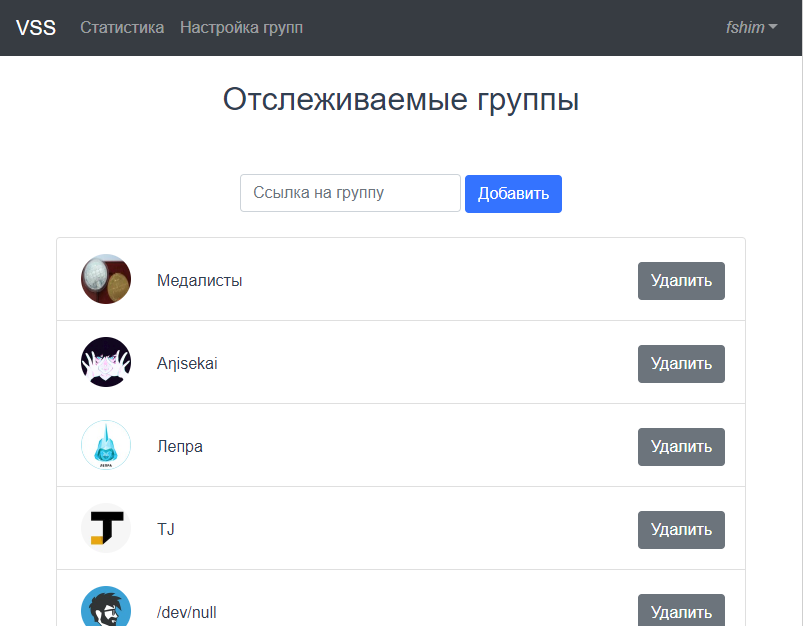


Рис. 2.20 - Отслеживаемые группы

Для добавления группы нужно ввести ссылку в соответствующее поле (Рис. 2.21) и нажать кнопку добавить.

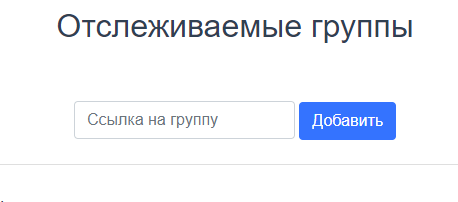


Рис. 2.21 – Добавление групп

Для удаления группы необходимо нажать на кнопку «Удалить», которая находится в каждой строке списка (рис. 2.22).

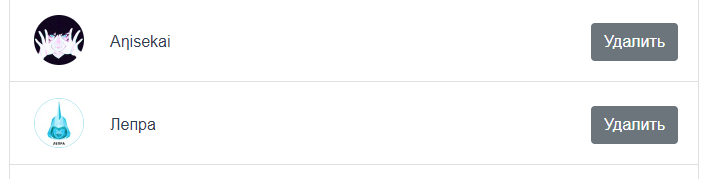


Рис. 2.22 – Удаление

## Тестирование приложения

Для корректной работы приложения необходимо, чтобы секретный ключ пользователя (тoкен) всегда был актуален, без него приложение не будет работать корректно, так как приложение постоянно делает запросы на получение информации у API VK.

Для этого был придуман механизм, при котором пользователь не сможет продолжить работу с приложение, пока не предоставит ключ.

Если у пользователя некорректный или не рабочий ключ, то при переходе на страницу статистики ему выведется сообщение (Рис. 2.23) о не настроенных группах.

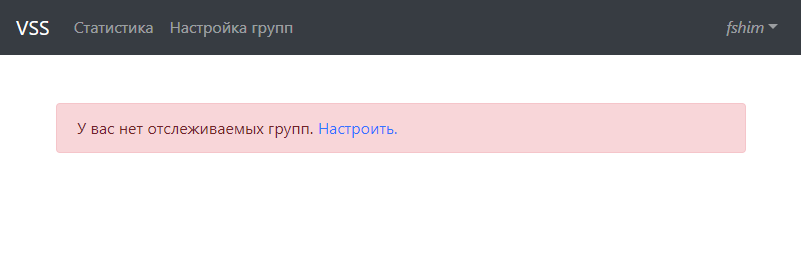


Рис. 2.23 – Сообщение

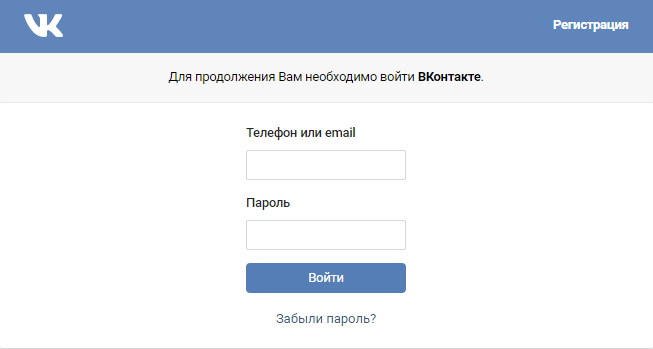
После перехода на страницу «Настройка групп» пользователю будет предложено авторизоваться (рисунок 2.24), если он не авторизован. 

Рис. 2.24 – Авторизация Вконтакте

Далее пользователю нужно будет предоставить разрешение на взаимодействие приложение с аккаунтом пользователя (Рис. 2.25).

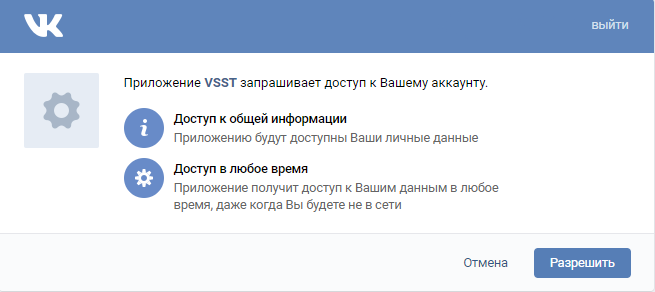


Рис. 2.25 - Окно

В таблице 2.1 представлены результаты тестирования добавления группы на старице настройки групп.

Таблица 2.1 – тестирование ввода пользователя

|  |  |
| --- | --- |
| Ввод с клавиатуры | Реакция программы |
| https://vk.com/oldlentach | Группа добавлена |
| oldlentach | Группа добавлена |
| 2343112 | Группа добавлена |
| Просто текст | Группа не добавлена |

## Руководство системного программиста

Рекомендации по минимальной аппаратной составляющей:

* Процессор: Intel(R) Xeon CPU E5-2690 2.90 GHz (4 ядра);
* Платформа: 32-х или 64-х разрядная;
* Оперативная память: Не менее 5 Gb;
* Жесткий диск: Не менее 2000 Mb свободного объема.

Поддерживаются следующие версии ОС семейства Linux:

* Debian;
* Ubuntu;
* Red Hat Enterprise Linux;
* Astra Linux.

В таблице 2.2 представлены команды, которые необходимо выполнить в терминале на сервере для корректной работы приложения.

Таблица 2.2 – Настройка сервера

|  |  |
| --- | --- |
| Команда | Комментарий |
| sudo apt install python3.6 | Установка python3 |
| pip3 install django | Установка django |
| sudo apt-get install apache2 | Установка Apache2 |
| sudo apt-get install libapache2-mod-wsgi | Установка mod\_wsgi |

Продолжение таблицы 2.2

|  |  |
| --- | --- |
| curl -sL https://deb.nodesource.com/setup\_8.x | sudo -E bash -  sudo apt-get install -y nodejs | Установка nodejs и npm |
| sudo apt-get install mysql-server  mysql\_secure\_installation | Установка MySQL |

## Руководство программиста

Для разработки пользовательского интерфейса была применена технология сборки Webpack. Webpack является одним из самых мощных и гибких инструментов для сборки frontend.

Webpack берет модули с зависимостями и генерирует статические ресурсы, которые представляют эти модули. Упрощенный механизм работы от разработчиков webpack можно увидеть на рисунке 2.26.

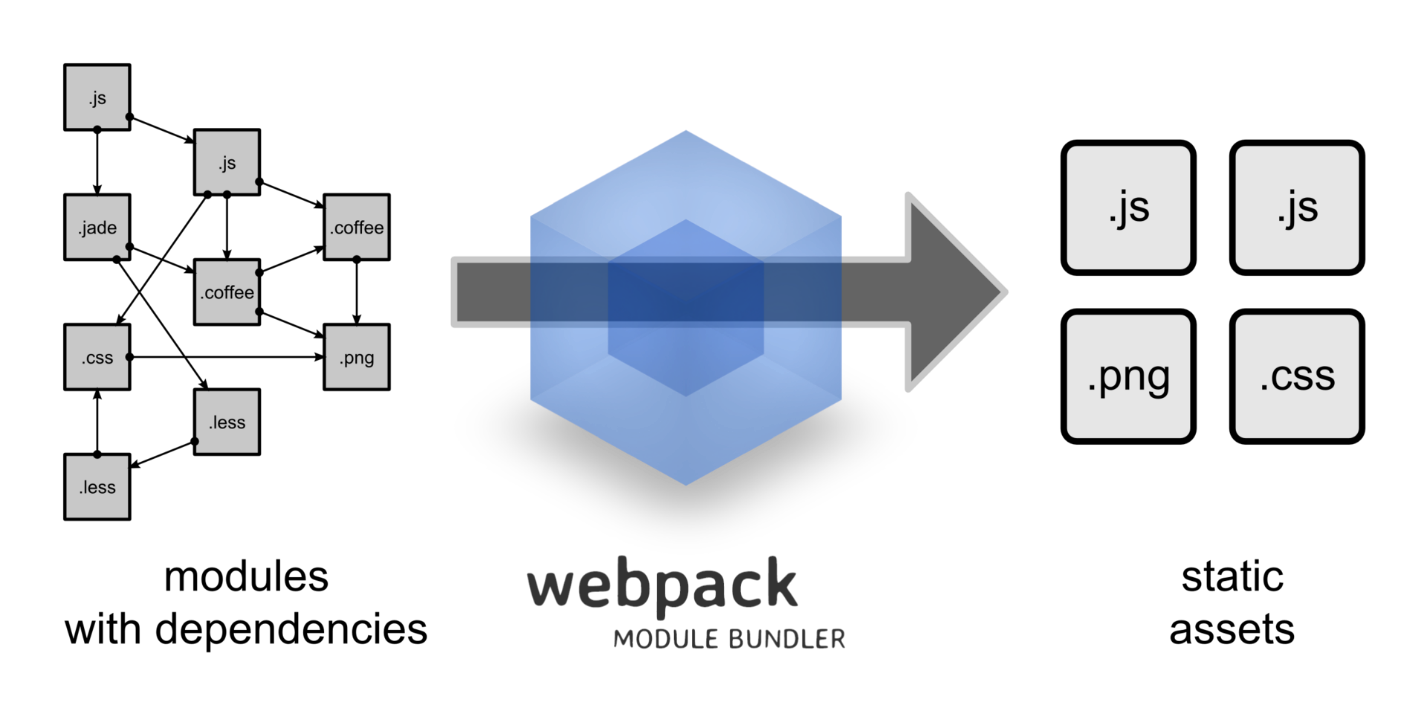


Рис. 2.26 – Механизм работы

Для стабильной работы рекомендуется использовать Linux подобную операционную систему. В таблице 2.3 приведены команды которые необходимо выполнить в терминале для установки webpack.

Таблица 2.3 – Установка webpack

|  |  |
| --- | --- |
| Команда | Комментарий |
| curl -sL https://deb.nodesource.com/setup\_8.x | sudo -E bash -  sudo apt-get install -y nodejs | Установка nodejs и npm |
| npm install webpack --save-dev | Установка webpack |

## Руководство пользователя

Для запуска программы у пользователя должен быть установлен веб-браузер с поддержкой JavaScript. Далее в списке приводится список рекомендуемых браузеров:

* Google Chrome;
* Mozilla Firefox;
* Opera;
* Яндекс.Браузер;
* Microsoft Edge.

Пользователю также необходимо иметь доступ интерне, либо находиться в одной локальной сети с сервером.

Для открытия приложения пользователь вводит в адресную строку ip-адрес или доменное имя приложения (в зависимости от настроек сервера). На рисунке 2.27 показан пример в браузере Mozilla Firefox.

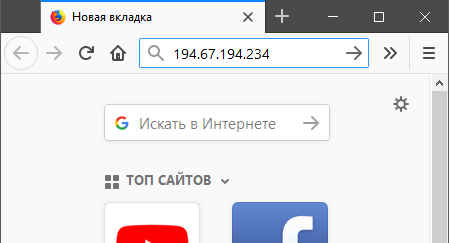


Рис. 2.27 – Пример

Далее нужно нажать на кнопку «Перейти», либо нажать клавишу «Enter».

# Заключение

Главная цель проекта была достигнута благодаря написанному современному, простому в использование, но в тоже время информативному приложению, которое помогло бы администраторам публичных страниц без особых усилий понять вектор их направления, в котором стоит развивать сообщество. Понятный интерфейс и прозрачная логика работы, чтобы приложением могли пользоваться как профессионалы, так и простые пользователи, например, художник, который публикует свои работы.

Одной из главных задач, стоявших перед программистом – это архитектура приложения. Разработана архитектура приложения, рассчитанная на большое количество пользователей с возможности расширения. Правильно настраивая сервера можно плавно перейти от проекта с посещаемостью 100 человек в сутки до 100000 человек в сутки и больше. Так же приложение предусматривает обработку и хранение «тяжелых» данных на отдельном сервере, что положительно влияет на систему в целом и на производительность в частности. Подводя итог сказанному, можно заключить, что задача была выполнена в полной мере.

Не менее важной задачей в проекте с большим количеством пользователей – это грамотно спроектированная база данных. Было разработано две базы данных для серверной части и вспомогательной части, что прекрасно вписывается в общую архитектуру приложения и дополняет ее.

В рамках задачи по подбору компонентов было необходимо подобрать компоненты так, чтобы они корректно работали в связке между собой и вписывались в общую архитектуру проекта. Так же необходимо учитывать лицензирование компонентов, так как это может накладывать свои ограничения и увеличить себестоимость приложения. Все компоненты, подобранные в проекте, удовлетворяют требованиям, имеют лицензию MIT, благодаря этому было сэкономлено время на разработку и соответственно уменьшило себестоимость проекта.

# Список используемой литературы

Законодательные и нормативные акты

1. ГОСТ 7.1. – 2003. Библиографическая запись. Библиографическое описание. Общие требования и правила составления. – М.: ИПК Издательство стандартов, 2004. – 169 с.
2. ГОСТ 7.32 – 2001. Система стандартов по информацию, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления. – М.: ИПК Издательство стандартов, 2001. – 21 с.
3. ГОСТ 7.82 – 2001. Библиографическая запись. Библиографическое описание электронных ресурсов. Общие требования и правила составления. – М.: ИПК Издательство стандартов, 2001. – 21 с.

Учебная и научная литература

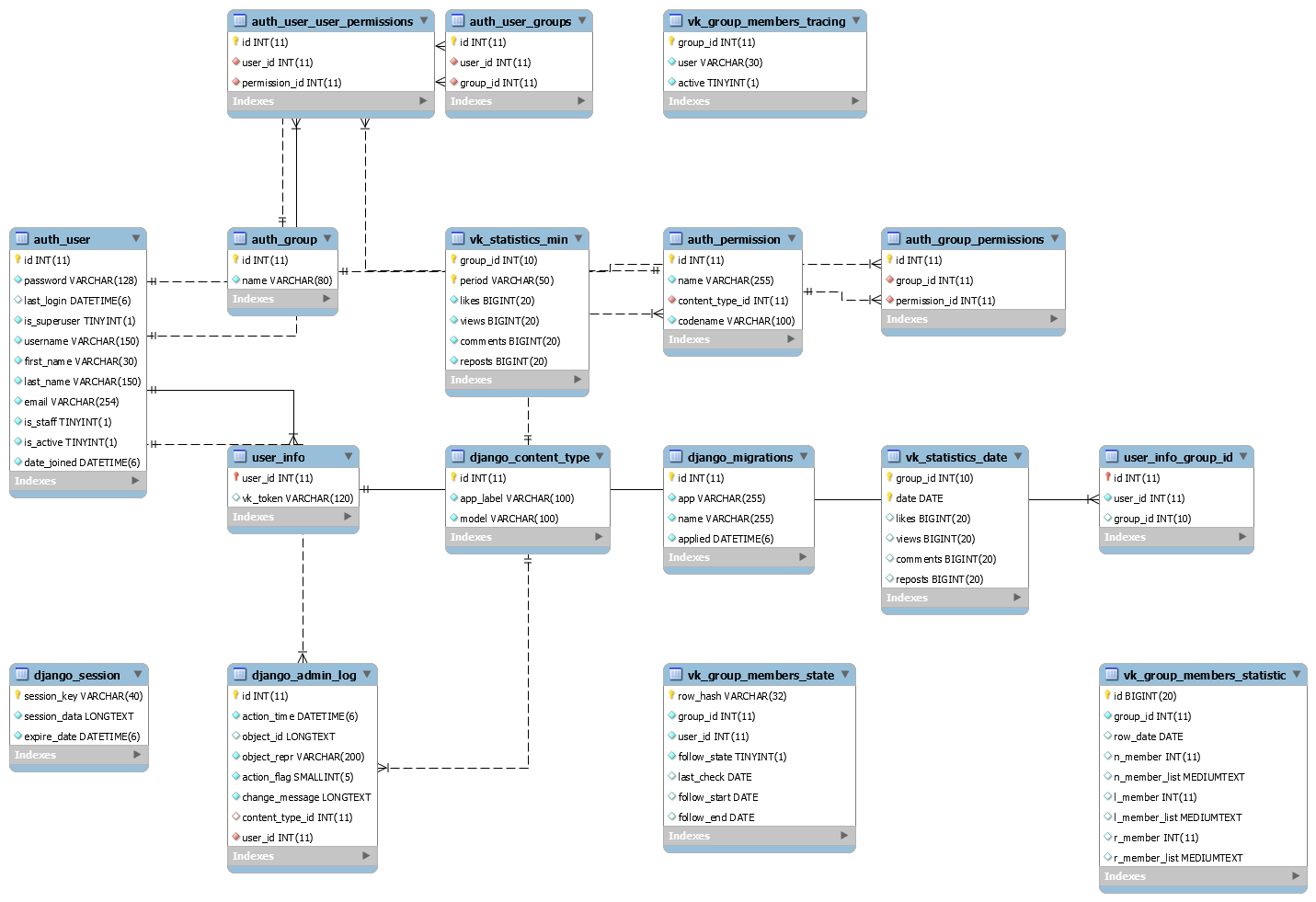
1. Кришна Г. Хороший интерфейс - невидимый интерфейс – М.: СПБ: Питер, 2016. – 256 с.
2. Павловская Т.А. С#. Программирование на языке высокого уровня: учебник для вузов. – СПб: Питер, 2016. – 432 с.
3. Майкл Н. Проектирование и дизайн ПО для тех, кому не все равно. – СПб.: Питер, 2016. – 320 с.

Интернет-документы

1. Bootstrap – Документация по Bootstrap [Электронный ресурс]. – Режим доступа: <http://getbootstrap.ru/docs/3.3.7/components/>.
2. CSS – Справочник по CSS [Электронный ресурс]. – Режим доступа: <http://htmlbook.ru/css>.
3. HTML – Справочник по HTML [Электронный ресурс]. – Режим доступа: <http://htmlbook.ru/html>.
4. JSON – Документация по JSON [Электронный ресурс]. – Режим доступа: https://learn.javascript.ru/json.
5. NodeJS – Англоязычная документация по NodeJS [Электронный ресурс]. – Режим доступа: <https://nodejs.org/en/docs/>.
6. REST – Описание REST в Википедии [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/REST>.
7. VKAPI – API для работы с социальной сетью «ВКонтакте» [Электронный ресурс]. – Режим доступа: <https://vk.com/dev>.
8. Django – Документация по Django [Электронный ресурс]. – Режим доступа: <https://docs.djangoproject.com/en/2.0/>.
9. VueJS – Документация по VueJS [Электронный ресурс]. – Режим доступа: <https://ru.vuejs.org/v2/guide/index.html>.
10. Python – Документация по Python [Электронный ресурс]. – Режим доступа: <https://docs.python.org/3/index.html>.
11. Webpack – Документация по webpack [Электронный ресурс]. – Режим доступа: https://webpack.js.org/concepts/.

# Приложение А

Диаграмма классов для базы данных.



# Приложение Б

Титульный лист презентации к выпускной квалификационной работы.

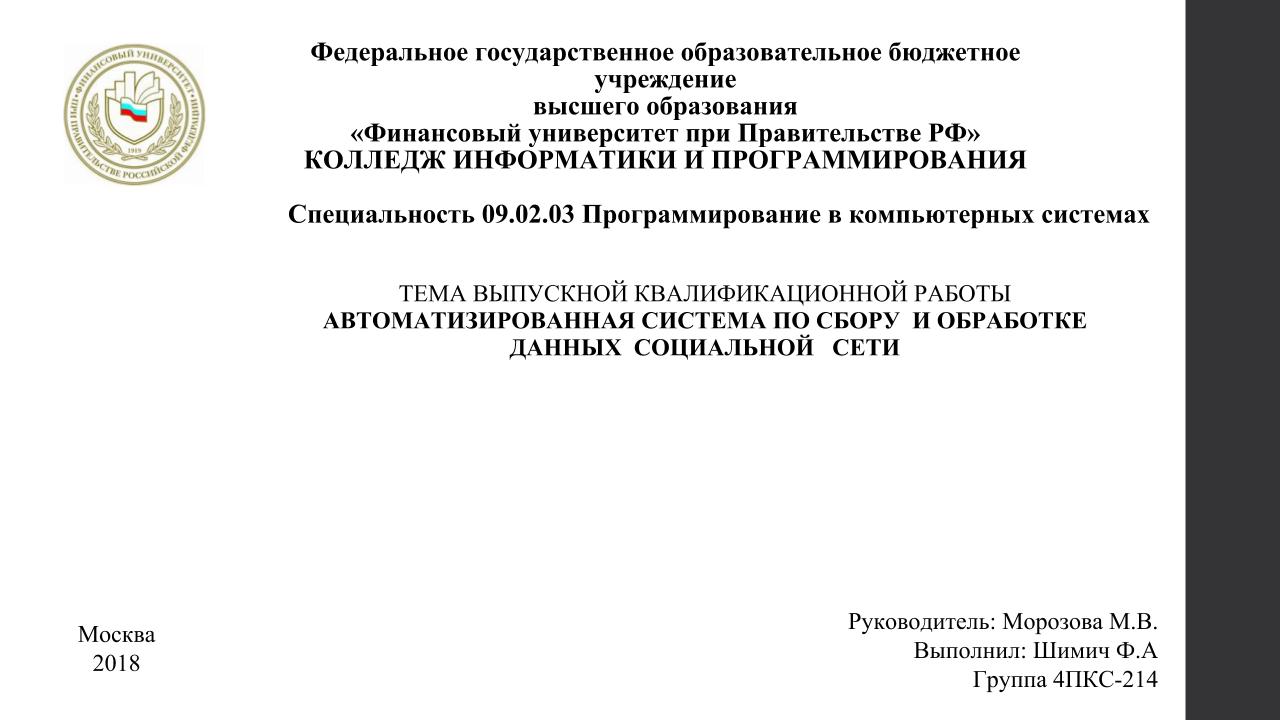


Рис. Б.1 - Титульный лист презентации к ВКР.

# Приложение В

Листинг App.vue

<template>

<div id="app">

<b-navbar toggleable="md" type="dark" variant="dark">

<b-navbar-toggle target="nav\_collapse"></b-navbar-toggle>

<b-navbar-brand href="">VSS</b-navbar-brand>

<b-collapse is-nav id="nav\_collapse">

<b-navbar-nav>

<b-nav-item href="/panel">Статистика</b-nav-item>

<b-nav-item href="panel/groups">Настройка групп</b-nav-item>

</b-navbar-nav>

<!-- Right aligned nav items -->

<b-navbar-nav class="ml-auto">

<b-nav-item-dropdown right>

<!-- Using button-content slot -->

<template slot="button-content">

<em>{{ user }}</em>

</template>

<b-dropdown-item href="#">Профиль</b-dropdown-item>

<b-dropdown-item href="/auth/logout">Выход</b-dropdown-item>

</b-nav-item-dropdown>

</b-navbar-nav>

</b-collapse>

</b-navbar>

<br><br>

<b-container class="bv-example-row">

<div v-if="groups.length === 1">

<b-row>

<b-col>

<b-alert show variant="danger">У вас нет отслеживаемых групп. <b-link href="/panel/groups">Настроить.</b-link></b-alert>

</b-col>

</b-row>

</div>

<div v-if="groups.length > 1">

<b-row>

<b-col>

<b-form-select v-model="selectedGroup" :options="groups" class="mb-3" />

</b-col>

<b-col cols="4">

<!--<b-img rounded="circle" blank width="75" height="75" blank-color="#777" alt="img" class="m-1" src="https://pp.userapi.com/c840329/v840329975/1df4/tEt8pa\_Qq9A.jpg"/> -->

<div v-if="selectedGroup != null"><b-button v-bind:href="'https://vk.com/public' + selectedGroup">Открыть в Вконтакте</b-button></div>

</b-col>

</b-row>

<b-row>

<b-col>

<b-form-select v-model="selectedView" :options="options" class="mb-3">

<option value="members">Статистика участников</option>

<option value="statistics-min">Статистика контента</option>

<option value="statistics-date">Статистика контента (подробная)</option>

<option value="post-vk">Посты</option>

<option value="post-fav">Посты (избранное)</option>

</b-form-select>

</b-col>

<b-col cols="4">

<b-button v-b-toggle.support class="m-1">Развернуть справку</b-button>

</b-col>

</b-row>

<members v-if="selectedView === 'members'" v-bind:group-id="selectedGroup"></members>

<statistics-min v-if="selectedView === 'statistics-min'" v-bind:group-id="selectedGroup"></statistics-min>

<statistics-date v-if="selectedView === 'statistics-date'" v-bind:group-id="selectedGroup"></statistics-date>

<posts-vk v-if="selectedView === 'post-vk'" v-bind:group-id="selectedGroup"></posts-vk>

<posts-fav v-if="selectedView === 'post-fav'"></posts-fav>

</div>

</b-container>

</div>

</template>

<script>

import Members from './components/Members'

import StatisticsMin from './components/Statistics-min'

import StatisticsDate from './components/Statistics-date'

import PostsVk from './components/Posts-vk'

import PostsFav from './components/Posts-fav'

import axios from 'axios'

import fetchJsonp from 'fetch-jsonp'

axios.defaults.xsrfCookieName = 'sessionid'

axios.defaults.xsrfCookieName = 'csrftoken'

axios.defaults.xsrfHeaderName = 'X-CSRFToken'

export default {

name: 'app',

components: {

Members,

StatisticsMin,

StatisticsDate,

PostsVk,

PostsFav

},

data() {

return{

selectedGroup: null,

selectedView: "members",

options: [],

groups: [{value: null, text: "Выберете группу"}],

user: \_user,

token: \_tocken

}

},

created() {

var self = this;

axios.get('/api/get\_groups')

.then(function (response){

if (response.data['response'] < 0){

return;

}

var items = response.data['response'];

/\*

for (var i in items) {

self.groups.push({value: items[i]['group\_id'], text: 'id: ' + items[i]['group\_id']});

}

\*/

self.getGroupsInfo(items);

//self.groups = response.data['response'];

})

.catch(function (error){

console.log(error);

});

},

methods: {

getGroupsInfo(items){

var self = this;

var url = 'https://api.vk.com/method/groups.getById?v=5.78&group\_ids=' + items.map(e => e.group\_id).join(",") + '&access\_token=' + this.token;

fetchJsonp(url)

.then(function(response) {

return response.json()

}).then(function(json) {

var response = json['response'];

var items = [{value: null, text: "Выберете группу"}];

for (var i in response) {

items.push({text: response[i]['name'], value: response[i]['id']});

}

self.groups = items;

}).catch(function(ex) {

console.log('parsing failed', ex)

})

}

}

}

</script>

Листинг DChart.js

import VueCharts from 'vue-chartjs'

import {Doughnut} from 'vue-chartjs'

export default {

extends: Doughnut,

props: ['data'],

watch: {

data: function() {

if (this.data != null){

this.setData();

}

}

},

mounted () {

if (this.data != null){

this.setData();

}

},

methods: {

setData: function(){

var datacollection = {

labels: ['Вступили', 'Вышли', 'Вернулись'],

datasets: [

{

backgroundColor: [

'#41B883',

'#E46651',

'#00D8FF'

],

data: this.data

}

]

};

this.renderChart(datacollection, {responsive: true, maintainAspectRatio: false});

}

}

}

Листинг Members.vue

<template>

<div class="members">

<b-row>

<b-col>

<b-collapse id="support" style="margin-bottom: 15px;">

<b-card>

I am collapsable content!

</b-card>

</b-collapse>

</b-col>

</b-row>

<b-row>

<b-col>

<b-alert :show="groupId == null">Группа не выбрана</b-alert>

</b-col>

</b-row>

<b-row v-if="groupId != null">

<b-col cols="4">

<b-form-group label="Период:">

<b-form-input type="date" name="start" v-model="date" value="date"></b-form-input>

</b-form-group>

</b-col>

</b-row>

<b-row>

<b-col>

<b-alert :show="Object.keys(members).length == 0 && groupId != null">Нет данных</b-alert>

</b-col>

</b-row>

<div v-if="Object.keys(members).length > 0">

<b-row>

<b-col>

<b-alert show variant="success">Вступило: {{ members.n\_member }}</b-alert>

<div id="scrollspy-nested" style="position:relative;height:350px;overflow-y:auto">

<b-list-group>

<b-list-group-item v-for="item in members.n\_member\_list" :key="item">{{ item }}</b-list-group-item>

</b-list-group>

</div>

</b-col>

<b-col>

<b-alert show variant="danger">Вышло: {{ members.l\_member }}</b-alert>

<div id="scrollspy-nested" style="position:relative;height:350px;overflow-y:auto">

<b-list-group>

<b-list-group-item v-for="item in members.l\_member\_list" :key="item">{{ item }}</b-list-group-item>

</b-list-group>

</div>

</b-col>

<b-col>

<b-alert show variant="dark">Вернулись: {{ members.r\_member }}</b-alert>

<div id="scrollspy-nested" style="position:relative;height:350px;overflow-y:auto">

<b-list-group>

<b-list-group-item v-for="item in members.r\_member\_list" :key="item">{{ item }}</b-list-group-item>

</b-list-group>

</div>

</b-col>

</b-row>

<b-row>

<b-col>

<d-chart :data="chart"><d-chart>

</b-col>

</b-row>

</div>

</div>

</template>

<script>

import axios from 'axios'

axios.defaults.xsrfCookieName = 'sessionid'

axios.defaults.xsrfCookieName = 'csrftoken'

axios.defaults.xsrfHeaderName = 'X-CSRFToken'

import VueCharts from 'vue-chartjs'

import { Bar, Line, Doughnut } from 'vue-chartjs'

import DChart from './DChart'

export default {

name: 'members',

components: {

DChart

},

props: ['groupId'],

watch: {

groupId: function(newVal) {

if(this.groupId != null){

this.setDataNow();

this.getMembers();

}

},

date: function(newVal) {

this.getMembers();

}

},

data () {

return{

group\_id: null,

members: {},

date: new Date(),

chart: null

}

},

created: function (){

if(this.groupId != null){

this.setDataNow();

this.getMembers();

}

},

methods: {

getMembers: function () {

var self = this;

axios.post('/api/get\_members', {

group\_id: self.groupId,

date: self.date

})

.then(function (response) {

if(response.data['status'] == 1){

var temp = response.data['response'];

temp['n\_member\_list'] = temp['n\_member\_list'] != null ? temp['n\_member\_list'].split(",") : null;

temp['l\_member\_list'] = temp['l\_member\_list'] != null ? temp['l\_member\_list'].split(",") : null;

temp['r\_member\_list'] = temp['r\_member\_list'] != null ? temp['r\_member\_list'].split(",") : null;

self.members = temp;

self.chart = [

self.members.n\_member,

self.members.l\_member,

self.members.r\_member

];

console.log(self.chart);

}

else {

self.members = {};

}

})

.catch(function (error) {

self.members = {};

console.log(error);

});

},

setDataNow: function() {

Date.prototype.yyyymmdd = function() {

var mm = this.getMonth() + 1;

var dd = this.getDate();

return [this.getFullYear(),

(mm>9 ? '' : '0') + mm,

(dd>9 ? '' : '0') + dd

].join('-');

};

var date = new Date();

this.date = date.yyyymmdd();

}

}

}

</script>

<!-- Add "scoped" attribute to limit CSS to this component only -->

<style scoped>

h1, h2 {

font-weight: normal;

}

ul {

list-style-type: none;

padding: 0;

}

li {

display: inline-block;

margin: 0 10px;

}

a {

color: #42b983;

}

.members{

margin-top: 20px;

}

</style>

Листинг models.py

from django.db import models

class AuthGroup(models.Model):

name = models.CharField(unique=True, max\_length=80)

class Meta:

managed = False

db\_table = 'auth\_group'

class AuthGroupPermissions(models.Model):

group = models.ForeignKey(AuthGroup, models.DO\_NOTHING)

permission = models.ForeignKey('AuthPermission', models.DO\_NOTHING)

class Meta:

managed = False

db\_table = 'auth\_group\_permissions'

unique\_together = (('group', 'permission'),)

class AuthPermission(models.Model):

name = models.CharField(max\_length=255)

content\_type = models.ForeignKey('DjangoContentType', models.DO\_NOTHING)

codename = models.CharField(max\_length=100)

class Meta:

managed = False

db\_table = 'auth\_permission'

unique\_together = (('content\_type', 'codename'),)

class AuthUser(models.Model):

password = models.CharField(max\_length=128)

last\_login = models.DateTimeField(blank=True, null=True)

is\_superuser = models.IntegerField()

username = models.CharField(unique=True, max\_length=150)

first\_name = models.CharField(max\_length=30)

last\_name = models.CharField(max\_length=150)

email = models.CharField(max\_length=254)

is\_staff = models.IntegerField()

is\_active = models.IntegerField()

date\_joined = models.DateTimeField()

class Meta:

managed = False

db\_table = 'auth\_user'

class AuthUserGroups(models.Model):

user = models.ForeignKey(AuthUser, models.DO\_NOTHING)

group = models.ForeignKey(AuthGroup, models.DO\_NOTHING)

class Meta:

managed = False

db\_table = 'auth\_user\_groups'

unique\_together = (('user', 'group'),)

class AuthUserUserPermissions(models.Model):

user = models.ForeignKey(AuthUser, models.DO\_NOTHING)

permission = models.ForeignKey(AuthPermission, models.DO\_NOTHING)

class Meta:

managed = False

db\_table = 'auth\_user\_user\_permissions'

unique\_together = (('user', 'permission'),)

class DjangoAdminLog(models.Model):

action\_time = models.DateTimeField()

object\_id = models.TextField(blank=True, null=True)

object\_repr = models.CharField(max\_length=200)

action\_flag = models.PositiveSmallIntegerField()

change\_message = models.TextField()

content\_type = models.ForeignKey('DjangoContentType', models.DO\_NOTHING, blank=True, null=True)

user = models.ForeignKey(AuthUser, models.DO\_NOTHING)

class Meta:

managed = False

db\_table = 'django\_admin\_log'

class DjangoContentType(models.Model):

app\_label = models.CharField(max\_length=100)

model = models.CharField(max\_length=100)

class Meta:

managed = False

db\_table = 'django\_content\_type'

unique\_together = (('app\_label', 'model'),)

class DjangoMigrations(models.Model):

app = models.CharField(max\_length=255)

name = models.CharField(max\_length=255)

applied = models.DateTimeField()

class Meta:

managed = False

db\_table = 'django\_migrations'

class DjangoSession(models.Model):

session\_key = models.CharField(primary\_key=True, max\_length=40)

session\_data = models.TextField()

expire\_date = models.DateTimeField()

class Meta:

managed = False

db\_table = 'django\_session'

class UserInfo(models.Model):

user = models.ForeignKey(AuthUser, models.DO\_NOTHING, primary\_key=True)

vk\_token = models.CharField(max\_length=120, blank=True, null=True)

class Meta:

managed = False

db\_table = 'user\_info'

class UserInfoGroupId(models.Model):

user\_id = models.IntegerField()

group\_id = models.PositiveIntegerField(blank=True, null=True)

class Meta:

managed = False

db\_table = 'user\_info\_group\_id'

class VkGroupMembersState(models.Model):

row\_hash = models.CharField(primary\_key=True, max\_length=32)

group\_id = models.IntegerField()

user\_id = models.IntegerField()

follow\_state = models.IntegerField()

last\_check = models.DateField(blank=True, null=True)

follow\_start = models.DateField(blank=True, null=True)

follow\_end = models.DateField(blank=True, null=True)

class Meta:

managed = False

db\_table = 'vk\_group\_members\_state'

class VkGroupMembersStatistic(models.Model):

id = models.BigIntegerField(primary\_key=True)

group\_id = models.IntegerField()

row\_date = models.DateField(blank=True, null=True)

n\_member = models.IntegerField(blank=True, null=True)

n\_member\_list = models.TextField(blank=True, null=True)

l\_member = models.IntegerField(blank=True, null=True)

l\_member\_list = models.TextField(blank=True, null=True)

r\_member = models.IntegerField(blank=True, null=True)

r\_member\_list = models.TextField(blank=True, null=True)

class Meta:

managed = False

db\_table = 'vk\_group\_members\_statistic'

class VkGroupMembersTracing(models.Model):

group\_id = models.IntegerField(primary\_key=True)

user = models.CharField(max\_length=30)

active = models.IntegerField()

class Meta:

managed = False

db\_table = 'vk\_group\_members\_tracing'

class VkStatisticsDate(models.Model):

group\_id = models.PositiveIntegerField(primary\_key=True)

date = models.DateField()

likes = models.BigIntegerField(blank=True, null=True)

views = models.BigIntegerField(blank=True, null=True)

comments = models.BigIntegerField(blank=True, null=True)

reposts = models.BigIntegerField(blank=True, null=True)

class Meta:

managed = False

db\_table = 'vk\_statistics\_date'

unique\_together = (('group\_id', 'date'),)

class VkStatisticsMin(models.Model):

group\_id = models.PositiveIntegerField(primary\_key=True)

period = models.CharField(max\_length=50)

likes = models.BigIntegerField()

views = models.BigIntegerField()

comments = models.BigIntegerField()

reposts = models.BigIntegerField()

class Meta:

managed = False

db\_table = 'vk\_statistics\_min'

unique\_together = (('group\_id', 'period'),)

Листинг Posts-vk.vue

<template>

<div class="posts-vk">

<b-row>

<b-col>

<b-collapse id="support" style="margin-bottom: 20px;">

<b-card>

После выбора фильтра выводятся посты у которых показатели выше среднего.

</b-card>

</b-collapse>

</b-col>

</b-row>

<b-row>

<b-col>

<b-alert :show="groupId == null">Группа не выбрана</b-alert>

</b-col>

</b-row>

<b-row v-if="groupId != null">

<b-col>

<b-form-group label="Фильтр:">

<b-form-checkbox-group id="checkboxes1" name="flavour1" v-model="selected" :options="options"></b-form-checkbox-group>

</b-form-group>

</b-col>

</b-row>

<b-row>

<b-col>

<b-list-group>

<b-list-group-item v-for="item in items" :key="item">

<wall :array="item" :group-id="groupId"></wall>

</b-list-group-item>

</b-list-group>

<!-- <wall v-for="item in items" :array="item"></wall> -->

</b-col>

</b-row>

</div>

</template>

<script>

import axios from 'axios'

import fetchJsonp from 'fetch-jsonp'

import Wall from './Wall'

export default {

name: 'posts-vk',

props: ['groupId'],

watch: {

groupId: function() {

if (this.groupId != null){

this.getItems()

}

else {

this.items = null;

}

},

selected: function() {

this.setFilter();

}

},

components: {

Wall

},

data () {

return {

err: false,

average: null,

\_items: [],

items: [],

token: \_token,

selected: [],

filter: [],

options: [

{text: 'Лайки', value: 'likes'},

{text: 'Просмотры', value: 'views'},

{text: 'Репосты', value: 'reposts'},

{text: 'Комментарии', value: 'comments'}

]

}

},

methods: {

getItems(){

var self = this;

var count = { likes: 0, views: 0, reposts: 0, comments: 0 };

var url = 'https://api.vk.com/method/wall.get?v=5.78&count=15&owner\_id=-' + this.groupId + '&access\_token=' + this.token;

fetchJsonp(url)

.then(function(response) {

return response.json()

}).then(function(json) {

self.\_items = json['response']['items'].filter(function(el) {

if('copy\_history' in el){

return false;

}

else{

count.likes += el.likes.count;

count.views += el.views.count;

count.reposts += el.reposts.count;

count.comments += el.comments.count;

return true;

}

});

self.average = {

likes: count.likes / self.\_items.length,

views: count.views / self.\_items.length,

reposts: count.reposts / self.\_items.length,

comments: count.comments / self.\_items.length

};

console.log(self.average);

self.items = self.\_items;

}).catch(function(ex) {

console.log('parsing failed', ex)

self.err = true;

self.items = null;

})

},

setFilter() {

if (this.selected.length === 0){

this.items = this.\_items;

return;

}

var self = this;

var el = this.\_items.filter(function(el) {

for (var i = 0; i < self.selected.length; i++) {

if (el[self.selected[i]].count < self.average[self.selected[i]]){

return false;

}

}

return true;

});

this.items = el;

}

},

created: function(){

if (this.groupId != null){

this.getItems()

}

}

}

</script>

<style>

#post-filter {

display: inline;

}

.posts-vk{

margin-top: 20px;

}

</style>

Листинг settings.py

import os

SECRET\_KEY = '’

DEBUG = False

ALLOWED\_HOSTS = ['194.67.194.234']

# Application definition

INSTALLED\_APPS = [

'login',

'vss',

'api',

'vk',

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

'django\_yuml',

]

MIDDLEWARE = [

'django.middleware.security.SecurityMiddleware',

'django.contrib.sessions.middleware.SessionMiddleware',

'django.middleware.common.CommonMiddleware',

'django.middleware.csrf.CsrfViewMiddleware',

'django.contrib.auth.middleware.AuthenticationMiddleware',

'django.contrib.messages.middleware.MessageMiddleware',

'django.middleware.clickjacking.XFrameOptionsMiddleware',

]

ROOT\_URLCONF = 'vss.urls'

TEMPLATES = [

{

'BACKEND': 'django.template.backends.django.DjangoTemplates',

'DIRS': [],

'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 = 'vss.wsgi.application'

# Database

# https://docs.djangoproject.com/en/2.0/ref/settings/#databases

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.mysql',

'NAME': 'vss',

'HOST': '127.0.0.1',

'PASSWORD': 'Zk.,k.vjkjrj6569',

'USER': 'root',

'PORT': '3306'

}

}

# Password validation

# https://docs.djangoproject.com/en/2.0/ref/settings/#auth-password-validators

AUTH\_PASSWORD\_VALIDATORS = [

{

'NAME': 'django.contrib.auth.password\_validation.UserAttributeSimilarityValidator',

},

{

'NAME': 'django.contrib.auth.password\_validation.MinimumLengthValidator',

},

{

'NAME': 'django.contrib.auth.password\_validation.CommonPasswordValidator',

},

{

'NAME': 'django.contrib.auth.password\_validation.NumericPasswordValidator',

},

]

# Internationalization

# https://docs.djangoproject.com/en/2.0/topics/i18n/

LANGUAGE\_CODE = 'en-us'

TIME\_ZONE = 'UTC'

USE\_I18N = True

USE\_L10N = True

USE\_TZ = True

APPEND\_SLASH=False

# Static files (CSS, JavaScript, Images)

# https://docs.djangoproject.com/en/2.0/howto/static-files/

STATIC\_URL = '/static/'

Листинг Statistics-date.vue

<template>

<div class="statistics-date">

<b-row>

<b-col>

<b-collapse id="support" style="margin-bottom: 20px;">

<b-card>

После выбора фильтра выводятся лучшие посты.

</b-card>

</b-collapse>

</b-col>

</b-row>

<b-row>

<b-col>

<b-alert :show="groupId == null">Группа не выбрана</b-alert>

</b-col>

</b-row>

<b-row>

<b-col>

<b-alert :show="items.length == 0 && groupId != null">Нет данных</b-alert>

</b-col>

</b-row>

<div v-if="items.length > 0">

<b-row>

<b-table striped hover :items="items" :fields="fields"></b-table>

</b-row>

</b-row>

<line-chart v-bind:data="items"/>

</b-row>

</div>

</div>

</template>

<script>

import axios from 'axios'

import VueCharts from 'vue-chartjs'

import { Bar, Line } from 'vue-chartjs'

import LineChart from './LineChart'

export default {

components: {

LineChart

},

name: 'statistics-date',

props: ['groupId'],

watch: {

groupId: function(newVal) {

if(this.groupId != null){

this.getStatistic();

}

}

},

data () {

return {

fields: [

{

key: 'date',

sortable: true,

label: "Дата"

},

{

key: 'views',

sortable: true,

label: "Просмотры"

},

{

key: 'likes',

label: 'Лайки',

sortable: true,

},

{

key: 'reposts',

label: 'Репосты',

sortable: true,

},

{

key: 'comments',

label: 'Комментарии',

sortable: true,

}

],

items: []

}

},

created: function (){

if(this.groupId != null){

this.getStatistic();

}

},

methods: {

getStatistic: function(){

var self = this;

axios.post('/api/get\_statistics\_date', {

group\_id: self.groupId,

})

.then(function (response) {

if(response.data['status'] == 1){

self.items = response.data['response'];

}

else {

self.items = [];

}

})

.catch(function (error) {

self.items = [];

console.log(error);

});

}

}

}

</script>

<style>

.statistics-date{

margin-top: 20px;

}

</style>

Листинг Statistics-min.vue

<template>

<div class="statistics-min">

<b-row>

<b-col>

<b-collapse id="support" style="margin-bottom: 20px;">

<b-card>

После выбора фильтра выводятся лучшие посты.

</b-card>

</b-collapse>

</b-col>

</b-row>

<b-row>

<b-col>

<b-alert :show="groupId == null">Группа не выбрана</b-alert>

</b-col>

</b-row>

<b-row>

<b-col>

<b-alert :show="items.length == 0 && groupId != null">Нет данных</b-alert>

</b-col>

</b-row>

<div v-if="items.length > 0">

<b-row>

<b-table striped hover :items="items" :fields="fields"></b-table>

</b-row>

</div>

</div>

</template>

<script>

import axios from 'axios'

export default {

name: 'statistics-min',

props: ['groupId'],

watch: {

groupId: function(newVal) {

if(this.groupId != null){

this.getStatistic();

}

}

},

data () {

return {

fields: [

{

key: 'period',

sortable: false,

label: "Период"

},

{

key: 'views',

sortable: false,

label: "Просмотры"

},

{

key: 'likes',

label: 'Лайки',

sortable: false,

},

{

key: 'reposts',

label: 'Репосты',

sortable: false,

},

{

key: 'comments',

label: 'Комментарии',

sortable: false,

}

],

items: []

}

},

created: function (){

if(this.groupId != null){

this.getStatistic();

}

},

methods: {

getStatistic: function(){

var self = this;

axios.post('/api/get\_statistics\_min', {

group\_id: self.groupId,

})

.then(function (response) {

if(response.data['status'] == 1){

self.items = response.data['response'];

}

else {

self.items = [];

}

})

.catch(function (error) {

self.items = [];

console.log(error);

});

}

}

}

</script>

<style>

.statistics-min{

margin-top: 20px;

}

</style>

Листинг urls.py

from django.contrib import admin

from django.urls import path, include

urlpatterns = [

path('admin/', admin.site.urls),

path('api', include('api.urls')),

path('panel', include('vk.urls')),

path('auth', include('login.urls')),

]

Листинг views-api.py

from django.shortcuts import render

from django.http import JsonResponse

from django.core.serializers import serialize

from datetime import datetime

import json

import time

from vss.models import VkGroupMembersTracing, VkGroupMembersStatistic, VkStatisticsMin, VkStatisticsDate, UserInfoGroupId, UserInfo

def del\_group(request):

data = json.loads(request.body)

group = UserInfoGroupId.objects.filter(group\_id = data['id'], user\_id = request.user.id)

group.delete()

return JsonResponse({'status' : 1})

def add\_group(request):

try:

if request.user.is\_authenticated:

data = json.loads(request.body)

item = UserInfoGroupId(user\_id = request.user.id, group\_id = data['group\_id'])

item.save()

return JsonResponse({'status' : 1, 'response': UserInfoGroupId.objects.order\_by('id').values()[0]})

else:

return JsonResponse({'status' : -1})

except:

return JsonResponse({'status' : -2})

def get\_groups(request):

try:

if request.user.is\_authenticated:

users = UserInfoGroupId.objects.filter(user\_id = request.user.id).values()

return JsonResponse({'response' : list(users)})

else:

return JsonResponse({'response' : -1})

except:

return JsonResponse({'response' : -2})

def index(request):

all\_o = VkGroupMembersTracing.objects.all().values\_list()

return JsonResponse({'response': list(all\_o)})

def get\_members\_statistics\_json(group\_id):

if group\_id is None:

return JsonResponse({'error' : 1})

ms = VkGroupMembersStatistic.objects.filter(group\_id = group\_id).values()

if ms is None or len(ms) == 0:

return JsonResponse({'error' : 0 })

else:

return JsonResponse(ms[0])

def get\_members\_statistics(request, group\_id = None):

#time.sleep(5)

if request.user.is\_authenticated:

return get\_members\_statistics\_json(group\_id)

else:

return JsonResponse({'error' : 2 })

def get\_statistics\_min(request):

if request.user.is\_authenticated:

data = json.loads(request.body)

item = VkStatisticsMin.objects.filter(group\_id = data['group\_id']).values()

if len(item) > 0:

return JsonResponse({'status' : 1, 'response': list(item) })

else:

return JsonResponse({'status' : 0})

else:

return JsonResponse({'status' : -1})

def get\_statistics\_date(request):

if request.user.is\_authenticated:

data = json.loads(request.body)

item = VkStatisticsDate.objects.filter(group\_id = data['group\_id']).values()

if len(item) > 0:

return JsonResponse({'status' : 1, 'response': list(item) })

else:

return JsonResponse({'status' : 0})

else:

return JsonResponse({'status' : -1})

def get\_members(request):

if request.user.is\_authenticated:

data = json.loads(request.body)

date = datetime.strptime(data['date'], '%Y-%m-%d').date()

item = VkGroupMembersStatistic.objects.filter(row\_date = date, group\_id = data['group\_id']).values()

if len(item) > 0:

return JsonResponse({'status' : 1, 'response': item[0]})

else:

return JsonResponse({'status' : 0})

else:

return JsonResponse({'status' : -1})

#-----VKAPI

import requests

def get\_groups\_info(request):

if request.user.is\_authenticated:

data = json.loads(request.body)

url = 'https://api.vk.com/method/groups.getById?group\_ids=' + '1234,4321' + '&access\_token=' + UserInfo.objects.get(user\_id = request.user.id).vk\_token

answer = requests.get(url=url).json()

return JsonResponse(answer)

else:

return JsonResponse({'status' : -1})

Листинг views-vk.py

from django.shortcuts import render, redirect

from django.contrib import auth

from vss.models import UserInfo

# Create your views here.

def login(request):

if request.POST:

username = request.POST.get('login', '')

password = request.POST.get('password', '')

user = auth.authenticate(username=username, password=password)

if user is not None:

auth.login(request, user)

return redirect('/panel')

else:

return render(request, 'login.html')

else:

return render(request, 'login.html')

def logout(request):

auth.logout(request);

return redirect('/auth/login')

import requests

import json

def set\_token(request):

code = request.GET.get('code', '')

url = 'https://oauth.vk.com/access\_token?client\_id=6465718&client\_secret=8ZizNjLNMl62I51rZ9e5&redirect\_uri=http://194.67.194.234/auth/vk&code=' + code

answer = requests.get(url=url).json()

if 'invalid\_grant' not in answer:

try:

info = UserInfo.objects.create(user\_id = request.user.id, vk\_token = answer['access\_token'])

except:

info = UserInfo.objects.get(user\_id = request.user.id)

info.vk\_token = answer['access\_token']

info.save()

return redirect('/panel/groups')

else:

return redirect('/panel/groups')

Листинг views-vss.py

from django.shortcuts import render, redirect

from vss.models import UserInfo

# Create your views here.

def index(request):

if request.user.is\_authenticated:

return render(request, 'index.html')

else:

return redirect('/auth/login')

def groups(request):

if request.user.is\_authenticated:

url = 'https://oauth.vk.com/authorize?client\_id=6465718&display=page&redirect\_uri=http://194.67.194.234/auth/vk&scope=offline&response\_type=code&v=5.78'

try:

info = UserInfo.objects.filter(user\_id = request.user.id).get()

if info.vk\_token is None:

return redirect(url)

except:

return redirect(url)

return render(request, 'group.html')

else:

return redirect('/auth/login')

Листинг Wall.vue

<template>

<div class="wall">

<p><b><b-link :href="'https://vk.com/wall-' + link" target="\_blank"> {{ date }} </b-link></b></p>

{{ text }}

<b-row>

<b-col>

<div id="p-wall">

<vue-picture-swipe :items="images" :options="{shareEl: false}"></vue-picture-swipe>

</div>

</b-col>

</b-row>

<b-row>

<b-col>

<div id="b-wall">

Лайки: {{ likes }} | Просмотры: {{ views }} | Репосты: {{ reposts }} | Комментарии: {{ comments }}

<div>

</b-col>

</b-row>

</div>

</template>

<script>

import axios from 'axios'

import VueGallery from 'vue-gallery';

import VuePictureSwipe from 'vue-picture-swipe';

export default {

name: 'wall',

props: ['array', 'groupId'],

watch: {

array: function(newVal) {

this.setWall();

}

},

data () {

return {

slide: 0,

images: [],

index: null,

date: null,

text: null,

likes: null,

views: null,

reposts: null,

comments: null,

link: null,

}

},

created: function (){

this.setWall();

},

methods: {

setWall: function(){

var item = this.array;

var att = this.array['attachments'];

//console.log(item);

for (var i in att) {

if(att[i]['type'] === 'photo'){

var sizes = att[i]['photo']['sizes'];

var len = att[i]['photo']['sizes'].length - 1;

this.images.push({

src: sizes[len - 1]['url'],

thumbnail: sizes[3]['url'],

w: sizes[len - 1]['width'],

h: sizes[len - 1]['height']

});

}

}

Date.prototype.yyyymmdd = function() {

var mm = this.getMonth() + 1; // getMonth() is zero-based

var dd = this.getDate();

return [

(dd>9 ? '' : '0') + dd,

(mm>9 ? '' : '0') + mm,

this.getFullYear()

].join('.') + ' ' +

[

this.getHours(),

this.getMinutes(),

].join(':');

};

var d = new Date(item['date'] \* 1000);

this.date = d.yyyymmdd();

this.text = item['text'];

this.likes = item.likes.count;

this.views = item.views.count;

this.reposts = item.reposts.count;

this.comments = item.comments.count;

this.link = this.groupId + '\_' + item.id;

}

},

components: {

'gallery': VueGallery,

VuePictureSwipe

}

}

</script>

<style>

figure > a > img {

float: left;

padding-right: 10px;

height: 300px;

}

#b-wall{

display: block;

width: 100%;

background-color: #F5F5F5;

border-radius: 5px;

margin-top: 20px;

padding: 15px;

}

#p-wall{

display: block;

width: 100%;

}

</style>