МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «КубГУ»)

Факультет компьютерных технологий и прикладной математики
Кафедра информационных технологий

КУРСОВАЯ РАБОТА

АНАЛИЗ ДАННЫХ ПОЛЬЗОВАТЕЛЕЙ ИЗ СОЦИАЛЬНОЙ СЕТИ ВКОНТАКТЕ

Работу выполнил _____Д.А. Вербицкий
(подпись)

Направление подготовки 02.03.03 «Математическое обеспечение и администрирование информационных систем» курс 3

Направленность (профиль) «Технология программирования»

Научный руководитель
ст. преп. _____А.А. Михайличенко
(подпись)

Нормоконтролер
канд. пед. наук, доц. _____А.В. Харченко
(подпись)

Краснодар
2025

СОДЕРЖАНИЕ

Введение 3

1 Социальные сети и данные пользователей 4

2 API vk.com 7

2.1 Описание vk.com API и его возможностей 7

2.2 Разделение прав доступа и их значения 10

2.3 Описание методов API, используемых для извлечения данных 14

Список использованных источников 32

ВВЕДЕНИЕ

Социальные сети являются одним из крупнейших источников данных о пользователях. Эти данные используются для анализа поведения, предпочтений, географического распределения пользователей, а также для разработки персонализированных приложений и рекламы. Сегодня компании активно используют информацию из социальных сетей для повышения качества своих сервисов и разработки новых продуктов.

Автоматизированный сбор данных с платформ, таких как Вконтакте, Facebook или Instagram, позволяет получать актуальную информацию о пользователях, их интересах, социальной активности. Эти данные могут быть полезны в исследованиях, маркетинге, а также при создании приложений, ориентированных на целевую аудиторию.

Целью курсовой работы является разработка программы, способной извлекать информацию о пользователях из социальной сети ВКонтакте и строить различные группы на их основе. Программа должна быть достаточно гибкой и безопасной, учитывать ограничения API и обеспечивать корректное хранение данных.

Для достижения поставленной цели необходимо выполнить следующие задачи:

– изучить специализированные методы для извлечения данных пользователей из социальных сетей;

– доработать алгоритм для извлечения данных с учетом требований безопасности и ограничений API;

– реализовать программу, которая будет анализировать и визуализировать полученные через API данные.

1 Социальные сети и данные пользователей

1.1 Понятие и классификация данных пользователей социальных сетей

Данные пользователей социальных сетей — это информация, которая отражает действия, предпочтения, характеристики и связи людей, зарегистрированных на социальных платформах. Эти данные могут включать как личную информацию о пользователях, так и их взаимодействие с другими людьми, участие в сообществах, публикации и активности.

Информация о пользователях социальных сетей используется для анализа поведения, предпочтений и взаимодействий в рамках сети. Эти данные помогают лучше понять потребности и интересы целевой аудитории, что позволяет создавать более персонализированные и релевантные продукты и услуги. В маркетинговых и исследовательских проектах данные пользователей также используются для определения трендов и прогнозирования поведения различных социальных групп.

Данные пользователей подлежат строгим правилам конфиденциальности и защите. Социальные сети устанавливают свои ограничения и условия использования данных, чтобы соблюсти принципы конфиденциальности и права пользователей. Например, закон GDPR в Европейском Союзе регулирует сбор, обработку и хранение данных пользователей, требуя от компаний предоставления информации о том, как данные используются.

1.2 Классификация данных пользователей

Рассмотрим личные данные, которые пользователи указывают при регистрации или настраивают в профиле. К этой категории относятся:

1) идентификационные данные: имя, фамилия, дата рождения, пол, ID

пользователя;

2) контактные данные: город проживания, страна, контактные телефоны и адреса электронной почты;

3) информация о работе и образовании: место учебы, специальность, профессия, текущее место работы;

4) дополнительные сведения: семейное положение, религиозные взгляды, политические предпочтения.

Рассмотрим социальные данные. Эти данные связаны с социальными связями и взаимодействиями пользователя на платформе. К этой категории относятся:

1) список друзей: информация о друзьях, которая может включать их идентификаторы, общее количество друзей и другие параметры;

2) подписки и группы: сообщества и группы, на которые подписан пользователь; эти данные позволяют понять интересы и увлечения пользователя;

3) подписчики: информация о тех, кто следит за активностью пользователя.

Рассмотрим контентные данные. Они связаны с публикациями и контентом, созданным пользователем. К этой категории относятся:

1) посты и публикации: текстовые записи, изображения и видео, размещенные пользователем;

2) комментарии: отклики на публикации других пользователей, а также на посты и фотографии;

3) лайки и реакции: информация о том, какие записи пользователь оценил, отметив их "лайком" или другой реакцией;

4) медиафайлы: фотографии, видео, аудиозаписи, которые пользователь загружает на свою страницу.

Рассмотрим поведенческие данные. Это информация, которая собирается на основе действий пользователя. К этой категории относятся:

1) история активности: даты входа в сеть, продолжительность сессий,

активные часы;

2) местоположение: если включено отслеживание местоположения, может быть доступна информация о местах, которые посещал пользователь;

3) интересы: данные, которые можно выделить на основе лайков, репостов, интересных пользователю групп и контента.

Рассмотрим демографические и статистические данные, которые собираются и агрегируются для анализа:

1) возрастные категории: распределение пользователей по возрасту;

2) пол: процентное соотношение мужчин и женщин;

3) география: распределение по регионам, странам и городам;

4) социально-экономический статус: данные, которые можно оценить по ряду факторов (например, место работы, уровень образования).

1.3 Применение классификации данных

Классификация данных может осуществляться для следующих целей:

1) в маркетинговых исследованиях данные классифицируются для создания профилей целевых аудиторий; например, зная возраст и интересы пользователей, можно запускать рекламу, ориентированную на конкретные демографические группы;

2) классификация данных позволяет создавать поведенческие модели, которые помогают прогнозировать активность пользователей; это может использоваться для улучшения пользовательского опыта в социальных сетях и построения рекомендаций;

3) данные социальных сетей используются в социологических и демографических исследованиях, где важно понимать, как распределяются пользователи по возрастным категориям, регионам и другим характеристикам.

2 API vk.com

2.1 Описание vk.com API и его возможностей

API (Application Programming Interface) социальной сети vk.com предоставляет разработчикам интерфейсы для взаимодействия с данными платформы и позволяет программно получать доступ к информации пользователей, управлять контентом и выполнять ряд других функций. Данный API включает множество методов и функций, которые упрощают интеграцию внешних приложений с социальной сетью, поддерживая как десктопные, так и мобильные платформы.

Общие характеристики vk.com API:

1) API VK является RESTful, что означает, что взаимодействие с ним осуществляется через HTTP-запросы, где методы запроса (GET, POST) определяют тип действия; результаты запросов возвращаются в формате JSON, что существенно упрощает обработку данных и их интеграцию в различные системы;

2) для доступа к данным и выполнению действий от имени пользователя требуется токен доступа (access token), получаемый через OAuth 2.0; Это обеспечивает безопасное взаимодействие с API, где уровень доступа зависит от предоставленных прав;

3) VK поддерживает несколько версий API, что позволяет разработчикам работать с конкретными возможностями и изменениями; указание версии API в запросе позволяет использовать определенные методы и функции и адаптироваться к обновлениям, которые вносятся в систему.

VK предоставляет доступ к данным пользователей. Одной из главных возможностей является доступ к профилям пользователей:

1) методы users.get, users.search — позволяют получать информацию о пользователях, включая основные сведения, список друзей, подписчиков и информацию о статусе активности;

2) методы friends.get и groups.get — дают доступ к списку друзей и сообществ, на которые подписан пользователь; эти методы позволяют построить социальную карту контактов и интересов.

VK API предоставляет широкий функционал для работы с контентом:

1) метод wall.get — используется для получения записей со стены пользователя, включая текстовые записи, изображения, видео и другую информацию, связанную с публикациями;

2) методы photos.get и video.get — дают доступ к фотоальбомам и видеофайлам, которые хранятся в профиле пользователя, и позволяют загружать и управлять медиа-контентом;

3) методы likes.getList, comments.get — предоставляют возможность получать данные о реакции аудитории на контент (лайки и комментарии).

Для взаимодействия с сообществами (группами, публичными страницами) VK API включает:

1) методы groups.getById, groups.getMembers — позволяют получить информацию о сообществе, его участниках, а также узнать количество подписчиков и их характеристики;

2) метод groups.search — используется для поиска сообществ по заданным критериям, что удобно для анализа интересов и предпочтений.

С помощью VK API можно организовать активное взаимодействие с пользователями:

1) метод messages.send — позволяет отправлять сообщения пользователям или в сообщества; он широко используется в чат-ботах, которые взаимодействуют с пользователями напрямую;

2) метод notifications.send — обеспечивает отправку уведомлений, которые пользователи видят в разделе уведомлений, что помогает привлечь внимание к контенту или событию;

3) методы polls.getById, polls.create — позволяют создавать и управлять опросами, собирая информацию о мнении аудитории по различным вопросам.

VK API предлагает инструменты для работы с рекламой:

1) методы ads.getStatistics, ads.createAds — позволяют получать статистику по рекламным кампаниям, создавать и настраивать объявления, управлять их отображением и оценивать результаты.

Для использования VK API необходимо учитывать его особенности использования:

1) VK API имеет лимиты на количество запросов в единицу времени (например, 3 запроса в секунду для обычных приложений); это важно учитывать при разработке, так как превышение лимита может привести к временной блокировке запросов;

2) для доступа к определенным данным необходимо запрашивать разрешения у пользователей; примеры разрешений: доступ к друзьям (friends), группам (groups), сообщениям (messages), публикациям на стене (wall) и т. д.; эти разрешения предоставляются через систему OAuth 2.0 и токены доступа, а также имеют временные ограничения;

3) VK API возвращает коды ошибок и сообщения в случае неверных запросов или отсутствия прав доступа; например, ошибка 5 обозначает отсутствие валидного токена доступа, а 10 — попытку обращения к закрытым данным; это позволяет программистам корректно обрабатывать исключительные ситуации и улучшить пользовательский опыт.

Примеры использования VK API в приложениях:

1) социальный анализ и исследования; с помощью VK API можно собирать данные для исследования социальных связей, активности и интересов пользователей; например, используя методы friends.get и groups.get, можно построить сеть взаимодействий и выявить ключевые связи в группах;

2) создание чат-ботов; для взаимодействия с пользователями VK API предлагает методы, связанные с отправкой сообщений, получением списка участников и ответов на сообщения; это позволяет создавать ботов для автоматизации общения, оповещения пользователей и проведения опросов;

3) аналитика и маркетинг; маркетологи используют данные VK API

для анализа целевой аудитории, исследования предпочтений и демографических характеристик, создания более точных рекламных кампаний; методы ads.getStatistics и users.get позволяют собирать информацию о взаимодействии пользователей с контентом и объявлениями;

4) развлекательные и образовательные приложения; VK API активно используется для создания интерактивных приложений, например, приложений для викторин, опросов и развлекательных конкурсов; эти приложения помогают удерживать интерес аудитории и взаимодействовать с ними.

Можно выделить следующие преимущества и недостатки VK API.

Преимущества использования VK API: VK API предоставляет разработчикам широкий функционал и гибкость, позволяя создавать разнообразные приложения, интегрированные с VK; доступ к демографическим, социальным и поведенческим данным открывает возможности для анализа аудитории и разработки персонализированных сервисов.

Ограничения использования VK API: использование VK API ограничено политикой конфиденциальности и правами доступа; кроме того, лимиты на частоту запросов требуют оптимизации работы с API, а несанкционированное использование данных может привести к блокировке приложения.

2.2 Разделение прав доступа и их значения

API социальной сети vk.com предоставляет доступ к широкому спектру данных и функций, однако доступ к этим данным регулируется системой прав доступа. Разделение прав доступа служит для обеспечения конфиденциальности и безопасности информации пользователей, ограничивая доступ к тем данным и функциям, которые могут использоваться только при явном согласии пользователя. Права доступа определяются в виде разрешений (scopes), которые запрашиваются приложением при авторизации

пользователя и влияют на тип данных, к которым это приложение получит доступ.

Основные права доступа VK API и их значение:

1) друзья(friends); доступ к списку друзей пользователя; с этим правом приложение может получать ID друзей, а также дополнительные данные профиля (например, имя, фамилия), если это разрешено настройками конфиденциальности; используется для анализа социальных связей и построения социальной сети контактов;

2) фото(photos); доступ к фотографиям пользователя; позволяет получать информацию о фотоальбомах, фотографиях и их метаданных (например, дата добавления, место съемки); это разрешение полезно для приложений, которые используют фотографии пользователя для анализа активности, создания фотоколлажей или личных альбомов;

3) аудио(audio); доступ к аудиозаписям пользователя; разрешает приложению получать список аудиофайлов, добавленных пользователем; используется для создания музыкальных приложений и рекомендаций;

4) видео(video); доступ к видеозаписям пользователя; с этим правом приложение может получать данные о видеофайлах, доступных в профиле, включая ссылки и метаданные; полезно для приложений, интегрирующих видеоконтент или анализирующих медиа-предпочтения пользователя;

5) истории(stories); доступ к историям пользователя и его друзей; позволяет приложению получать информацию о созданных историях, но это право ограничено и доступно только для некоторых приложений;

6) стена(wall); доступ к публикациям на стене пользователя; позволяет приложению получать записи, которые пользователь разместил на своей стене, включая текст, изображения, ссылки и видео; может использоваться для анализа контента, популярности записей и реакции аудитории;

7) сообщества(groups); доступ к группам, на которые подписан пользователь; Приложение может получать данные о сообществах пользователя, включая названия, тематику и статус (например, открытое или

закрытое сообщество); это право используется для анализа интересов пользователя и для создания приложений, связанных с сообществами;

8) почта(email); позволяет получить электронный адрес пользователя, указанный при регистрации; это право редко используется и требует отдельного согласия пользователя, так как считается конфиденциальной информацией;

9) уведомления(notifications); доступ для отправки уведомлений пользователю; Приложение с этим разрешением может отправлять оповещения, которые пользователь увидит в своей ленте уведомлений; это право используется для привлечения внимания к важной информации, новой активности или событиям в приложении;

10) документы(docs); доступ к документам пользователя; приложение может получать список документов, загруженных пользователем, что может быть полезно для приложений, которые работают с обменом файлами;

11) сообщения(messages); доступ к сообщениям пользователя; приложение с этим правом может отправлять и получать сообщения от имени пользователя или группы; данный доступ используется для создания чат-ботов, которые взаимодействуют с пользователями, отправляя сообщения с информацией, поддержкой или уведомлениями;

12) реклама(ads); доступ к рекламному кабинету пользователя; Предоставляет возможность управления рекламными кампаниями, получения статистики и настроек; это право используется для приложений, которые помогают анализировать эффективность рекламы и управлять объявлениями;

13) офлайн(offline); дает возможность приложению работать с данными пользователя даже в том случае, если он не находится в сети; обычно токены доступа ограничены по времени, но с этим правом приложение может действовать дольше, что удобно для фоновой аналитики и сбора данных.

Механизм предоставления прав доступа Права доступа (scopes) запрашиваются при авторизации пользователя с помощью протокола OAuth 2.0. Пользователь видит, какие права запрашивает приложение, и должен явно

подтвердить согласие, что обеспечивает прозрачность использования его данных. Примерный процесс включает в себя несколько шагов.

Запрос авторизации — приложение направляет пользователя на страницу авторизации VK, где запрашивает необходимые права доступа.

Подтверждение пользователя — пользователь видит список запрашиваемых прав и принимает решение о предоставлении доступа. Это дает пользователю контроль над своими данными.

Получение access token — после подтверждения пользователем доступа приложение получает токен, который позволяет выполнять запросы к API в пределах разрешенных прав.

Значение разделения прав доступа для безопасности и конфиденциальности.

Разделение прав доступа имеет важное значение для защиты личных данных и конфиденциальности пользователей, поскольку позволяет:

Ограничить доступ к данным — приложение может получить только ту информацию, которая действительно необходима для его работы, минимизируя риски злоупотребления данными.

Снизить риски утечек — ограничение прав доступа уменьшает вероятность утечки конфиденциальной информации, если токен доступа окажется скомпрометирован.

Повысить доверие к приложениям — пользователи могут видеть, какие именно данные запрашивает приложение, и давать согласие только на те права, которые они считают необходимыми.

Контроль за доступом — пользователь может в любой момент отозвать предоставленные права через настройки своего профиля, что обеспечивает дополнительную защиту.

Ограничения и особенности использования прав доступа Важно отметить, что VK API накладывает ограничения на доступ к определенным данным даже при наличии прав:

1) частные профили и закрытые данные — если у пользователя

профиль закрыт или данные защищены настройками конфиденциальности, приложение не сможет получить к ним доступ, даже если права были предоставлены;

2) лимиты на частоту запросов — каждое приложение имеет ограничения по частоте запросов, и превышение этих лимитов может привести к временной блокировке доступа;

3) срок действия токена доступа — токен, полученный с правом offline, действует дольше, чем обычный токен, однако пользователь может его отозвать, что завершит доступ приложения.

2.3 Описание методов API, используемых для извлечения данных

В социальной сети vk.com реализован обширный API, который предоставляет доступ к данным пользователей, их социальным связям и контенту. Методы API классифицированы по группам в зависимости от типа данных, к которым они предоставляют доступ. Рассмотрим ключевые методы, которые можно использовать для извлечения данных из VK, а также их функциональные возможности и параметры.

Рассмотрим методы для извлечения информации о пользователях

users.get(user_ids, fields, name_case)

Основной метод для получения информации о пользователе по его ID.

Параметры:

– user_ids — список идентификаторов пользователей (можно указать несколько ID, разделяя их запятыми);

– fields — список дополнительных полей, которые нужно получить (например, дата рождения, город, страна, контакты, информация о работе и образовании);

– name_case — склонение имени и фамилии (именительный, родительный, дательный падежи).

Описание: возвращает базовые данные профиля пользователя (имя,

фамилию, ID) и указанные дополнительные поля, такие как дата рождения, информация о работе и учебе, пол и др. Пример использования: метод полезен для получения основной информации о пользователях и создания баз данных профилей для дальнейшего анализа.

users.search(q, fields, age_from, age_to, city, country, sex, status, has_photo)

Рассмотрим метод для поиска пользователей по заданным критериям.

Параметры:

– q — поисковый запрос (имя, фамилия или часть имени);

– fields — дополнительные данные, которые нужно включить в результаты (например, интересы, место работы);

– age_from, age_to — диапазон возрастов;

– city, country — ID города и страны;

– sex — пол пользователя;

– status, has_photo — фильтры по статусу и наличию фотографии.

Описание: возвращает список пользователей, соответствующих заданным критериям поиска. Это позволяет выбирать пользователей по интересам, возрасту, месту жительства и другим параметрам. Пример использования: используется для фильтрации и поиска пользователей с определенными характеристиками.

Рассмотрим методы для извлечения данных о друзьях и подписчиках.

friends.get(user_id, order, fields, count, offset)

Метод для получения списка друзей пользователя.

Параметры:

– user_id — ID пользователя, для которого нужно получить список друзей;

– order — порядок сортировки (например, по популярности или случайный порядок);

– fields — дополнительные поля данных друзей (например, дата рождения, город, пол);

– count, offset — количество возвращаемых результатов и смещение для пагинации.

Описание: возвращает ID и основные данные друзей пользователя. Это позволяет строить социальную сеть контактов, определять наиболее популярные связи. Пример использования: используется для анализа социальной сети пользователя и построения графа его друзей.

users.getFollowers(user_id, fields, count, offset)

Рассмотрим метод для получения списка подписчиков пользователя.

Параметры:

– user_id — ID пользователя, для которого нужно получить список подписчиков;

– fields — дополнительные поля, которые нужно вернуть (например, пол, город);

– count, offset — количество возвращаемых результатов и смещение.

Описание: возвращает список пользователей, которые подписаны на данный профиль, что позволяет анализировать аудиторию и популярность. Пример использования: применяется для получения информации о фолловерах и анализа структуры подписчиков.

Рассмотрим методы для работы с сообществами и группами.

groups.get(user_id, extended, fields)

Метод для получения списка сообществ, в которых состоит пользователь.

Параметры:

– user_id — ID пользователя, для которого нужно получить список сообществ;

– extended — если значение равно 1, метод возвращает подробную информацию о каждом сообществе;

– fields — дополнительные поля для сообществ (например, описание, тематика).

Описание: возвращает ID и базовую информацию о сообществах, к которым принадлежит пользователь. Это помогает понять интересы и предпочтения пользователя. Пример использования: полезен для построения профиля интересов и анализа тематики сообществ, на которые подписан пользователь.

groups.getMembers(group_id, sort, fields, count, offset)

Рассмотрим метод для получения списка участников группы.

Параметры:

– group_id — ID группы или короткое имя сообщества;

– sort — порядок сортировки (по возрастанию ID или случайный порядок);

– fields — дополнительные поля, которые нужно вернуть (например, город, дата рождения участников).

Описание: возвращает ID и данные участников группы, что позволяет исследовать структуру аудитории и демографические данные. Пример использования: применяется для анализа состава участников группы и определения активной аудитории.

Рассмотрим методы для извлечения контентных данных.

wall.get(owner_id, count, offset, filter)

Метод для получения записей со стены пользователя или сообщества.

Параметры:

– owner_id — ID пользователя или сообщества, чьи записи нужно получить;

– count, offset — количество возвращаемых записей и смещение;

– filter — фильтрация записей (например, только владельца или всех пользователей).

Описание: возвращает публикации на стене, включая текст, медиафайлы и временные метки. Позволяет анализировать активность и предпочтения пользователя. Пример использования: используется для сбора данных о контенте, который публикует пользователь, и анализа его интересов и

активности.

photos.get(owner_id, album_id, rev, count, offset)

Рассмотрим метод для получения списка фотографий пользователя или сообщества.

Параметры:

– owner_id — ID владельца фотографий;

– album_id — ID альбома, из которого нужно извлечь фотографии (например, профильные фото или загруженные);

– rev — порядок сортировки фотографий (по дате загрузки);

– count, offset — количество возвращаемых фотографий и смещение.

Описание: возвращает список фотографий и метаданные (например, дата создания, количество лайков). Эти данные можно использовать для визуализации и анализа визуальных предпочтений пользователя. Пример использования: применяется для извлечения медиаданных, связанных с профилем пользователя.

Рассмотрим методы для получения данных о реакции пользователей.

likes.getList(type, owner_id, item_id, count, offset)

Метод для получения списка пользователей, которым понравился определенный контент (пост, фотография, видео).

Параметры:

– type — тип контента (публикация, фото, видео);

– owner_id — ID владельца контента;

– item_id — ID элемента (например, ID поста или фотографии);

– count, offset — количество возвращаемых записей и смещение.

Описание: возвращает список пользователей, которые поставили «лайк» контенту, что позволяет определить популярность контента и вовлеченность аудитории. Пример использования: используется для оценки реакций пользователей на контент и анализа их предпочтений.

comments.get(owner_id, post_id, count, offset)

Рассмотрим метод для получения списка комментариев к контенту (например, публикации, фотографии).

Параметры:

– owner_id — ID владельца контента;

– post_id — ID публикации, для которой нужно получить комментарии;

– count, offset — количество возвращаемых записей и смещение.

Описание: возвращает список комментариев, что позволяет анализировать мнение и реакцию аудитории на контент. Пример использования: применяется для анализа комментариев, что дает представление об отношении пользователей к контенту.

Рассмотрим методы для анализа активности пользователей.

status.get(user_id)

Метод для получения статуса пользователя.

Параметры:

– user_id — ID пользователя, чей статус нужно получить.

Описание: возвращает текст статуса пользователя, который может дать представление о его текущем настроении или интересах. Пример использования: используется для мониторинга изменений в статусе и определения текущих интересов пользователя.

account.getProfileInfo()

Метод для получения дополнительной информации о профиле пользователя, например, настройках конфиденциальности и базовых данных.

Параметры отсутствуют.

Описание: возвращает информацию о профиле, включая личные данные и настройки. Пример использования: позволяет получить дополнительную информацию для построения профиля пользователя.

3 Описание программы и анализ данных

3.1 Описание процесса извлечения и обработки данных

Для реализации программного продукта была выбрана СУБД SQLite и язык программирования Python.

Обработка данных перед их сохранением в базе данных является важным этапом, обеспечивающим целостность, структурированность и готовность информации для последующего анализа. В рамках курсовой работы процесс обработки данных из социальной сети vk.com следующие шаги:

Рассмотрим извлечение данных через API.

Данные пользователей и сообществ извлекаются с помощью методов API vk.com, таких как users.get, friends.get, users.getSubscriptions и groups.getById. Эти методы возвращают информацию в формате JSON, содержащую необработанные данные, полученные напрямую из социальной сети.

На этом этапе основное внимание уделяется корректному выполнению запросов:

– проверяется наличие необходимых прав доступа (токен авторизации с соответствующими разрешениями);

– обрабатываются ответы API, включая ситуации с ошибками (например, отсутствием данных, блокировкой аккаунтов или ошибками ограничения запросов).

Далее проводится валидация данных. Перед сохранением данных важно убедиться, что они соответствуют требованиям целевой базы данных:

– проверяется наличие обязательных полей, таких как vk_id, first_name, last_name;

– проверяются типы данных (например, vk_id должен быть числом, а

даты – в корректном формате);

– удостоверяется, что данные не содержат ошибок, вызванных неполнотой информации, некорректным форматом или пустыми значениями;

– если данные не проходят валидацию, они либо исключаются из обработки, либо приводятся к допустимому формату, если это возможно.

API vk.com возвращает данные в плоском JSON-формате, который может содержать избыточные или дублирующиеся данные. Для соответствия структуре базы данных выполняется нормализация:

– данные разделяются по логическим сущностям: пользователи, группы, связи между ними (сущности User, Group и UserGroup в базе данных);

– строки, содержащие массивы вложенных данных (например, интересы, родственники, подписки), преобразуются в удобный для хранения и обработки вид;

– даты (например, дата рождения) приводятся к унифицированному формату.

Далее проводится фильтрация данных. На данном этапе выполняется фильтрация данных для исключения ненужной или конфиденциальной информации:

– удаляются поля, которые не будут использоваться в дальнейшем анализе (например, временные данные, уникальные только для внутренней обработки);

– данные закрытых профилей пользователей, помеченные атрибутами is_closed или is_private, обрабатываются в соответствии с политикой конфиденциальности; такие записи могут либо исключаться из базы, либо сохраняться с ограниченным набором доступных полей.

Далее происходит обогащение данных. Оно подразумевает добавление дополнительной информации или расчет производных метрик. Например:

– определение возраста пользователя на основе его даты рождения;

– подсчет количества друзей или подписчиков;

– обработка данных о последнем визите (last_seen) для определения активности пользователя;

– подготовка данных к сохранению

– после нормализации и фильтрации данные преобразуются в формат, совместимый с целевой структурой базы данных;

– генерируются объекты ORM, представляющие сущности User, Group и UserGroup (используется SQLAlchemy);

– уникальные записи проверяются на наличие в базе данных для предотвращения дублирования.

– данные разбиваются на батчи, чтобы минимизировать нагрузку на базу данных при массовой вставке.

Затем происходит обработка ошибок и логирование. На всех этапах предусмотрены механизмы обработки ошибок:

– ошибки API (например, превышение лимитов запросов) логируются для дальнейшего анализа;

– некорректные или пропущенные данные фиксируются в отдельном логе;

– обработка данных продолжается даже при возникновении ошибок в отдельных записях.

Рассмотрим итоговый процесс.

Все описанные шаги обработки реализуются в соответствующих классах проекта, таких как UserManager и GroupManager. Например:

– метод get_info из класса UserManager отвечает за извлечение и нормализацию данных пользователей;

– класс UserService обрабатывает подготовленные данные перед сохранением их в базу.

Таким образом, процесс обработки данных гарантирует, что вся информация, поступающая в базу данных, будет соответствовать требованиям

структуры, иметь минимальное количество ошибок и быть готовой к дальнейшему использованию.

3.2 Схема сохранения данных в базе (ORM-модели)

Для структурированного хранения данных, извлеченных из социальной сети vk.com, используется реляционная база данных. Для взаимодействия с ней применяются ORM-модели, реализованные с использованием библиотеки SQLAlchemy. Данная схема позволяет описывать структуру базы данных на уровне классов Python, что делает код удобным и читаемым, а также упрощает операции с данными.

Для хранения данных разработана следующая схема, включающая три основные таблицы (рисунок 1):

1) users – таблица для хранения данных о пользователях (рисунок 2);
2) groups – таблица для хранения данных о группах (рисунок 3);
3) user_groups – таблица для хранения связей между пользователями и группами, которые они посещают или в которых состоят (рисунок 4).

Эти таблицы связаны между собой отношением многие-ко-многим.

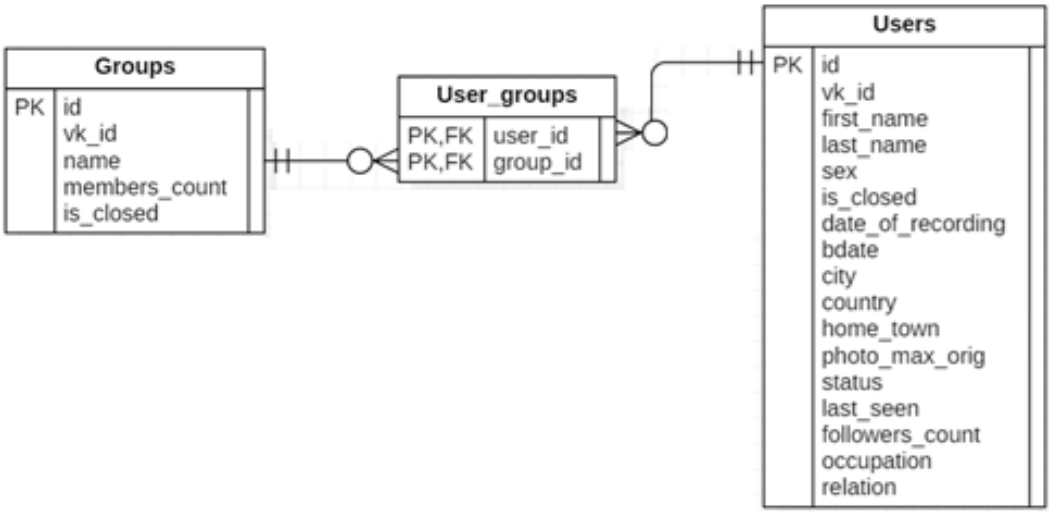

Рисунок 1 – ER диаграмма

Рассмотрим описание моделей.

Модели описаны в виде классов, которые наследуются от базового класса DeclarativeBase. Каждая модель соответствует одной таблице в базе данных.

Модель User представляет пользователей социальной сети. Включает как обязательные поля (например, ID), так и дополнительные (например, информация о городе, дате рождения).

Модель Group описывает сообщества (группы) в социальной сети.

Таблица users и таблица groups соединены через промежуточную таблицу user_groups.

Связь "пользователь ↔ группа" реализована через отношение многие-ко-многим.

Каждая запись в таблице user_groups связывает конкретного пользователя с конкретной группой.

	id	vk_id	first_name	last_name	sex	is_closed	date_of_recording	bdate	city	country	
1	1	3	DELETED		0	0	2024-11-22	<null>	<null>	<null>	<n
2	2	4	DELETED		0	0	2024-11-22	<null>	<null>	<null>	<n
3	3	5	Ilya	Perekopsky	2	0	2024-11-22	18.11	Moscow	<null>	<n
4	4	8	Aki	Sepiashvili	2	0	2024-11-22	<null>	Kyiv	<null>	<n
5	5	11	Mikhail	Petrov	2	0	2024-11-22	18.12	Saint Petersburg	<null>	Ca
6	6	14	Andrey	Gorodetsky	2	0	2024-11-22	<null>	London	<null>	Ле
7	7	17	Alexander	Bespalov	2	0	2024-11-22	21.10.1982	Saint Petersburg	<null>	<n
8	8	21	Mikhail	Ravdonikas	2	0	2024-11-22	23.7.1985	Saint Petersburg	<null>	
9	9	27	Yulia	Semikolenova	1	0	2024-11-22	9.1	Saint Petersburg	<null>	
10	10	39	David	Mirelli	2	0	2024-11-22	31.3	Tel Aviv	<null>	Пи
11	11	41	Vladislav	Miller	2	0	2024-11-22	3.2.1986	Moscow	<null>	<n
12	12	45	Sergey	Sergey	2	0	2024-11-22	28.1	Saint Petersburg	<null>	<n
13	13	47	Gabriel	Shalel	2	0	2024-11-22	30.7.1989	Moscow	<null>	<n
14	14	48	Misha	Melnikov	2	0	2024-11-22	<null>	Moscow	<null>	<n
15	15	56	Andrey	Chernyshev	2	0	2024-11-22	13.5.1985	Moscow	<null>	Мо
16	16	61	Grigory	Konradt	2	0	2024-11-22	9.3.1984	London	<null>	<n
17	17	65	Mikhail	Torline	2	0	2024-11-22	20.7	Moscow	<null>	<n
18	18	68	Vika	Mirilashvili	1	0	2024-11-22	12.5.1983	Saint Petersburg	<null>	<n
19	19	75	Ekaterina	Kopylova	1	0	2024-11-22	26.8.1988	Saint Petersburg	<null>	<n
20	20	79	David	Spekter	2	0	2024-11-22	<null>	Saint Petersburg	<null>	<n

Рисунок 2 – Таблица users

id	vk_id	name	members_count	is_closed
1	186931683	Kotov	24113	0
2	211273600	Seakomnata	3165	0
3	30936477	Street Styler | мода	1077059	0
4	55074079	Современная мама	1647839	0
5	108468	Кинопоиск	2815989	0
6	2158488	LIVE	4340035	0
7	9170627	smrt bk [smart book]	1051	0
8	15548215	ВЕДОМОСТИ	726448	0
9	15755094	РИА Новости	3240800	0
10	17862264	Аргументы и Факты / а…	367009	0
11	20537665	Роем!	157405	0
12	22222333	анекдотов.net	4513820	0
13	22884714	Команда Поддержки ВКо…	1298473	0
14	23064236	Четкие Приколы	6618864	0
15	23791221	TAdviser	5535	0
16	23863253	Смешные анекдоты	2130699	0
17	23956131	The Brown Room	21575	0
18	24199209	LIFE.ru	2608540	0
19	27060808	Интересные идеи Handm…	943186	0

Рисунок 3 – Таблица groups

user_id	group_id
3	2158488
3	11982368
3	12680100
3	15106510
3	15548215
3	15755094
3	20537665
3	21697953
3	22822305
3	23482909
3	23553134
3	23791221
3	23859803
3	24203916
3	24432306
3	25634363
3	26956082
3	28261334

Рисунок 4 – Таблица user_groups

3.3 Описание основного скрипта и анализа данных

Основной скрипт является центральным элементом проекта, который объединяет все модули и классы для реализации автоматизированного процесса извлечения данных из социальной сети VK, их обработки и сохранения в базе данных. Рассмотрим подробно структуру и ключевые этапы выполнения, логику работы основного скрипта.

Инициализация классов и настроек: подготовка необходимых объектов для работы с API, базой данных и сервисами.

Извлечение данных пользователей: получение информации о пользователях, их друзьях, подписках и группах.

Обработка данных: преобразование и фильтрация данных перед их сохранением.

Сохранение данных: запись извлеченной информации в базу данных с использованием ORM-моделей.

Контроль выполнения: ограничение объема извлекаемых данных и обработка ошибок для обеспечения устойчивости программы.

Рассмотрим структуру основного скрипта.

Основной скрипт состоит из следующих блоков:

1) импорты и настройки; на начальном этапе подключаются необходимые модули и устанавливаются константы, включая лимиты для предотвращения превышения допустимого объема данных, извлекаемых через API;

2) инициализация объектов; создаются экземпляры основных классов для взаимодействия с API и базой данных; эти объекты используются для выполнения ключевых операций, включая извлечение данных, их обработку и сохранение;

3) циклический процесс извлечения данных; основной процесс извлечения данных организован в виде вложенных циклов; внешний цикл

обрабатывает пользователей партиями, а внутренний цикл извлекает данные для каждого пользователя, такие как подписки на группы и их идентификаторы;

4) сохранение данных в базу; извлеченные данные обрабатываются и сохраняются в соответствующие таблицы базы данных; для этого используются методы сервисов, предназначенных для работы с пользователями, группами и их связями;

5) контроль объема данных; для предотвращения перегрузки API и базы данных реализовано ограничение на общее количество обрабатываемых пользователей; после достижения заданного лимита процесс извлечения данных завершается.

Рассмотрим алгоритм работы.

Основной скрипт работает по следующему алгоритму:

– инициализация необходимых объектов и установка лимитов на объем извлекаемых данных;

– формирование списка идентификаторов пользователей;

– извлечение данных о подписках пользователей с помощью менеджеров API;

– сохранение информации о пользователях, группах и их связях в базе данных;

– проверка на достижение лимита обработанных пользователей;

– завершение работы после обработки заданного количества данных.

Рассмотрим преимущества используемой структуры.

Модульность: код разбит на отдельные классы, что упрощает его тестирование и модификацию.

Эффективность: пакетная обработка данных снижает количество запросов к API.

Автоматизация: все операции, от извлечения до сохранения данных, выполняются без участия пользователя.

Устойчивость: обработка ошибок и контроль лимитов предотвращают

сбои при работе с API и базой данных.

В рамках курсовой работы был реализован аналитический модуль группировки пользователей, который расширяет функциональность основного скрипта. Этот модуль автоматически сегментирует пользовательскую базу на группы по демографическим и поведенческим характеристикам, используя алгоритм K-means. После загрузки данных из SQLite модуль выполняет предварительную обработку: заполняет пропуски, преобразует категориальные признаки (пол, город, семейное положение) и масштабирует числовые показатели (возраст, количество подписчиков, активность).

Модуль сочетает различные методы анализа данных, включая комбинированную обработку числовых и категориальных признаков через ColumnTransformer. Особое внимание уделено визуализации результатов: программа автоматически генерирует точечные диаграммы, отображающие распределение пользователей по возрасту и количеству подписчиков; по последней активности и количеству пользователей; по городам.

Интеграция модуля с основным скриптом позволяет не только собирать и хранить данные пользователей, но и получать подробные аналитические выводы о структуре аудитории. Использование методов машинного обучения дает возможность выявлять скрытые закономерности и естественные группы в пользовательской базе. Автоматизированная визуализация значительно упрощает интерпретацию результатов и помогает быстро оценить особенности каждого сегмента.

В итоге в процессе работы были исследованы 26000 страниц пользователей VK. Были извлечены основные данные и характеристики пользователей и помещены в соответствующие модели (рисунок 2). Также были извлечены данные о группах, в которых состоят пользователи (рисунок 3), и создана таблица, которая хранит подписки пользователей (рисунок 4). На основе собранных данных были проведены группировки пользователей, которые выявили сегменты аудитории с характерными

демографическими и поведенческими особенностями. Визуализация результатов группировок наглядно демонстрирует распределение пользователей по количеству подписчиков и возрасту (рисунок 5), активности (рисунок 6), городам (рисунок 7), что позволяет эффективно сегментировать аудиторию для маркетинговых задач.

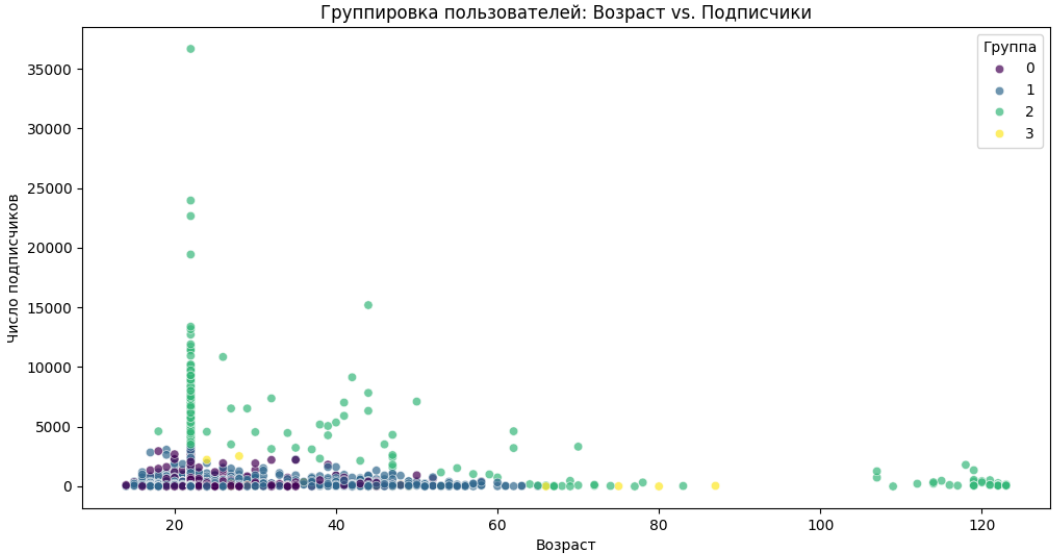

Рисунок 5 – Группировка пользователей по числу подписчиков и возрасту

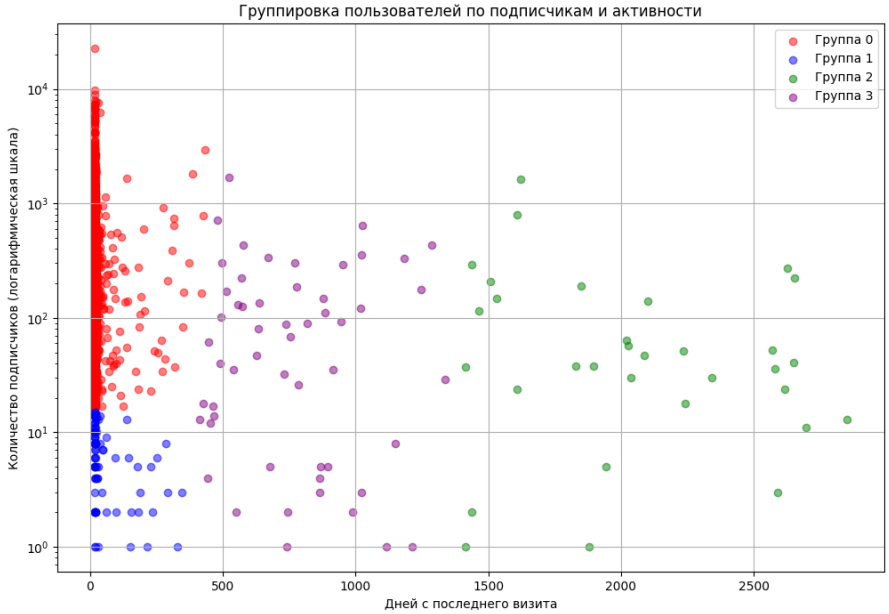

Рисунок 6 – Группировка пользователей по числу подписчиков и активности

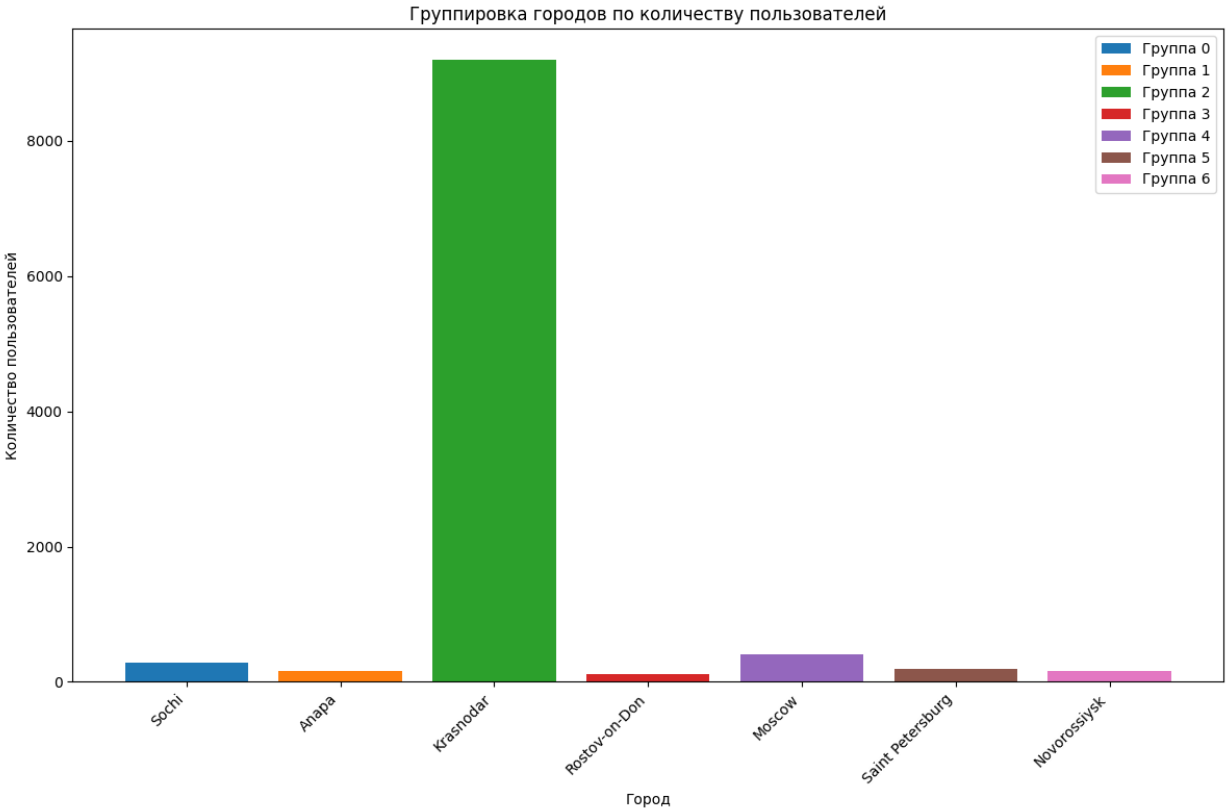

Рисунок 7 – Группировка пользователей по городам

ЗАКЛЮЧЕНИЕ

Цель курсовой работы, заключающаяся в разработке программы, способной извлекать информацию о пользователях из социальной сети ВКонтакте и строить различные группы на их основе, достигнута.

В ходе курсовой работы был доработан алгоритм для извлечения данных с учетом требований безопасности и ограничений API, модифицирована программа, которая запрашивает данные через API, сохраняет информацию в базу данных и строит графики некоторых групп на ее основе.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. VK API Documentation // Официальный сайт разработчиков VK: [сайт]. – 2024. – URL: https://dev.vk.com/ (дата обращения: 10.04.2025).

2. SQLAlchemy Documentation // Документация библиотеки SQLAlchemy: [сайт]. – 2024. – URL: https://docs.sqlalchemy.org/ (дата обращения: 26.04.2025).

3. Python Official Documentation // Официальная документация Python: [сайт]. – 2024. – URL: https://docs.python.org/ (дата обращения: 12.05.2025).

4. Гасанов, И.З. Эффективная работа с данными сообществ на примере api вконтакте/ И.З Гасанов // Инновации и инвестиции. – 2023. – №1. – URL: https://cyberleninka.ru/article/n/effektivnaya-rabota-s-dannymi-soobschestv-na-primere-api-vkontakte (дата обращения: 26.04.2025).