УДК 004.896

Разработка системы агрегации новостей сообществ ВКонтакте с использованием Flask и VK API

Малик Александр Игоревич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В статье рассматривается процесс разработки системы агрегации новостей сообществ социальных сетей, ориентированной на платформу ВКонтакте. Целью исследования является создание веб-приложения, которое позволяет пользователям получать актуальные новости из различных сообществ в одном месте. В статье описаны используемые технологии, методы интеграции с VK API и реализация веб-приложения на основе фреймворка Flask. Также проведен анализ существующих аналогов и представлены результаты разработки, включая дизайн интерфейса и функциональные возможности системы.
Keywords: система агрегации новостей, ВКонтакте, VK API, Flask, веб-приложение, социальные сети

Development of a News Aggregation System for VK Communities Using Flask and VK API

Malik Aleksandr Igorevich
Sholom-Aleichem Priamursky State University
Student

Abstract
The article examines the process of developing a news aggregation system for social network communities, specifically targeting the VK platform. The aim of the research is to create a web application that allows users to receive up-to-date news from various communities in one place. The article describes the technologies used, methods of integration with the VK API, and the implementation of the web application based on the Flask framework. An analysis of existing analogs is provided, and the development results, including interface design and system functionality, are presented.
Keywords: news aggregation system, VK, VK API, Flask, web application, social networks

1 Введение

1.1 Актуальность

С развитием социальных сетей и увеличением объема информации, распространяемой в них, возникает потребность в эффективных системах

агрегации новостей. Такие системы позволяют пользователям экономить время и получать актуальную информацию из различных источников в одном месте. ВКонтакте, являясь одной из крупнейших социальных сетей в России, предоставляет API для взаимодействия с данными пользователей и сообществ, что делает возможным создание специализированных новостных агрегаторов. Разработка системы агрегации новостей для ВКонтакте актуальна для пользователей, заинтересованных в оперативном получении новостей из различных сообществ.

1.2 Обзор исследований

В своей бакалаврской работе Ж. Мусашен из Университета Твенте разработал систему агрегирования и валидации новостей, исследуя существующие работы и литературу по этому вопросу. Основное внимание было уделено проектированию пользовательского интерфейса и методам проверки достоверности новостей. После разработки концепции и реализации системы, проведена оценка через создание трех сценариев, опрос и тестирование удобства использования. Результаты показали, что система в целом положительно воспринимается, хотя обратная связь указывает на возможность улучшения пользовательского опыта через более детализированный подход [1]. В статье Д. А. Кирьянова, опубликованной в журнале "Программные системы и вычислительные методы", рассматриваются методы построения систем агрегации контента, включая их архитектуру, масштабируемость и производительность. Исследование сосредоточено на разработке высокоуровневой архитектуры для распределенных высоконагруженных систем, автоматизирующих сбор информации из различных источников. В статье подробно анализируются принципы работы агрегаторов контента, такие как веб-кроулинг, обнаружение нечетких дубликатов, суммаризация данных и политика обновления. Автор предлагает рекомендации по выбору архитектурных стилей и программного обеспечения, таких как системы управления распределенными базами данных и брокеры сообщений. Основной вывод исследования — необходимость подхода к построению систем агрегации контента с учетом принципов распределенных систем, разделяя их на три логические части: агрегация, обработка данных и представление информации [2]. В статье A. Kamble, A. Gagan, P. Wagh, U. Aziz и Dr. M. R. Rajput описывают разработку веб-приложения для автоматического суммирования новостей. Целью данного приложения является предоставление читателям кратких резюме новостей, которые позволяют быстро уловить основные моменты статьи без необходимости чтения всего текста. Система использует веб-краулер для сбора информации с местных онлайн-ресурсов и применяет API суммирования для создания резюме. В статье комбинируются абстрактивные и экстрактивные методы суммирования для улучшения точности и качества выводимого контента. Для реализации фронтенда использован Streamlit, а для бэкенда – веб-фреймворк Flask. В проекте также применяются дополнительные зависимости Python, такие как Pillow и urllib.

Основные ключевые слова включают AWS, API, Streamlit, Anaconda, Newspaper 3k, Flask, Pillow, Aylien, Text Analysis и Meaning Cloud [3]. В статье авторы Р. К. Рай, Др. И. Сингх, А. Мудия и К. Бишт представляют веб-приложение для агрегации новостей, разработанное на основе фреймворка Django. Приложение предназначено для сбора и отображения новостных статей с различных веб-сайтов в одном месте, что позволяет пользователям легко и быстро получить актуальные новости. Система использует веб-скрапинг для извлечения информации с выбранных новостных сайтов и агрегирует её, чтобы пользователи могли просматривать статьи без необходимости посещать несколько сайтов. Приложение включает функционал для фильтрации и сортировки контента по пользовательским критериям, а также предоставляет API для создания пользовательских приложений. Одной из особенностей является отсутствие рекламы, что улучшает пользовательский опыт. Авторы также отмечают, что приложение упрощает доступ к новостям и снижает время, затрачиваемое на их поиск, благодаря объединению информации из нескольких источников [4]. В статье Г. Ханджи, А. Б. Менон, Р. Акшай, А. Патил и Б. Нандеани разрабатывают приложение для веб-скрапинга и анализа настроений, использующее фреймворк Flask. Основной целью проекта является создание системы, которая может извлекать текстовые отзывы с сайтов электронной коммерции и новостных блогов, а также проводить их анализ для определения положительных или отрицательных эмоций пользователей. Система позволяет пользователям вводить URL, выбирать, что именно нужно скрапировать (новости или отзывы), и затем применяет методы анализа настроений, чтобы определить отношение авторов к продукту или новости. В приложении используются инструменты, такие как Selenium для веб-скрапинга и NLTK для анализа текста, а также алгоритм TextRank для извлечения ключевых аспектов из отзывов [5].

1.3 Цель исследования

Целью данного исследования является создание системы, которая позволит эффективно собирать и предоставлять пользователю актуальные новости из различных сообществ социальной сети ВКонтакте.

2 Материалы и методы

В ходе исследования проводился теоретический анализ информации по теме агрегации новостей и интеграции с социальными сетями. Изучались различные методы сбора и обработки данных, а также подходы к созданию удобных пользовательских интерфейсов. На основе проведенного анализа были выбраны инструменты для разработки системы.

Для создания веб-приложения использовался фреймворк Flask на языке программирования Python. Flask был выбран за его простоту, гибкость и возможность быстрой разработки. Взаимодействие с социальной сетью ВКонтакте обеспечивалось с помощью VK API, что позволило получать

данные о новостях из сообществ. Были изучены и применены методы работы с API, включая аутентификацию, получение и обработку данных.

Интерфейс пользователя разрабатывался с использованием HTML, CSS и Bootstrap. Это позволило создать удобный и интуитивно понятный интерфейс, соответствующий современным требованиям веб-дизайна. Были реализованы элементы управления, позволяющие пользователям вводить идентификаторы сообществ и просматривать полученные новости в удобном формате.

Проводилось тестирование и отладка системы для обеспечения корректной работы. Тестирование включало проверку на различных устройствах и в разных браузерах, что позволило выявить и устранить возможные ошибки. Также была проведена работа по оптимизации производительности системы, что обеспечило стабильную и быструю работу приложения.

3 Результаты и обсуждения

3.1 Установка зависимостей

Начнем создание агрегатора с установки зависимостей. Так, как будет использоваться фреймворк Flask и взаимодействие с сообществами социальной сети Вконтакте – начнем с установки этих зависимостей (Рис. 1).

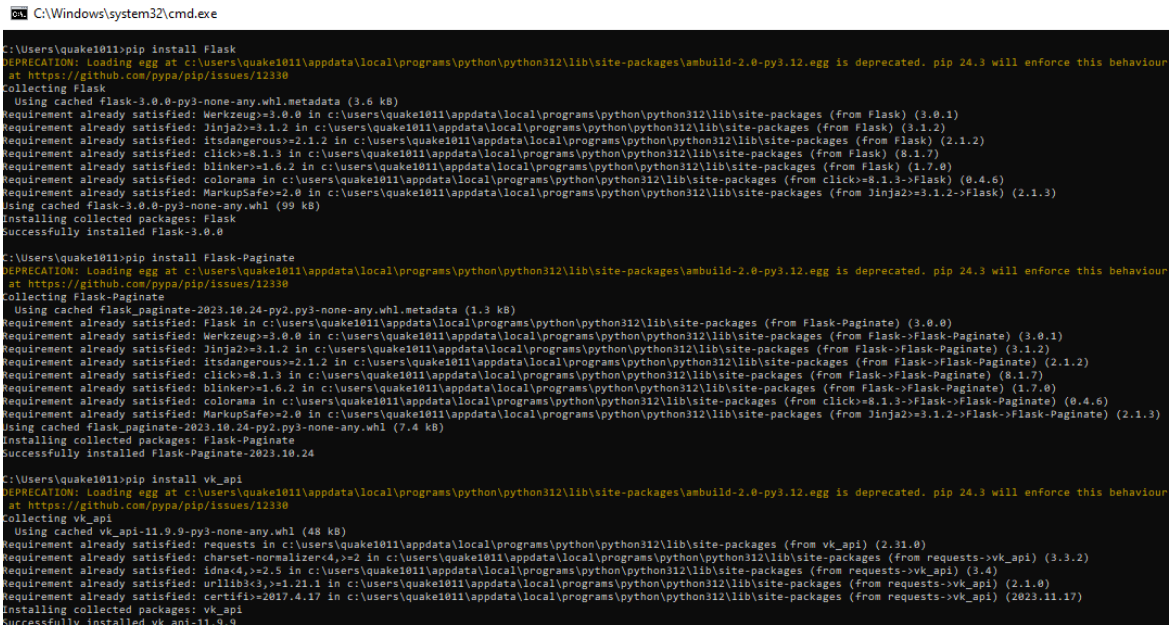

Рисунок 1. Установка зависимостей через командную строку Windows, используя пакетный установщик Python

Таким образом, устанавливаем Flask, Flask-Paginate для работы с пагинацией и библиотеку vk_api для взаимодействия с VK API.

3.2 Настройка VK API

Далее необходимо осуществить подготовку к взаимодействию с VK API. Для этого нужно иметь действующий токен, который предоставляет доступ к необходимым методам API.

Получение токена **VK API**:

1. Переход на страницу создания приложений VK.
2. Нажатие на "Создать приложение" или "Создать сообщество", в зависимости от потребностей.
3. Ввод названия приложения и выбор типа (standalone для личного использования, сайт, сторонние или Android/iOS).
4. После создания приложения скопировать его ID из адресной строки браузера.

Получение **access_token**:

1. Необходимо вернутся на страницу созданного приложения в Мои приложения.
2. Перейти во вкладку "Настройки" и найти раздел "Настройки". Там можно увидеть "Сервисный ключ доступа". Далее необходимо скопировать его.
3. Теперь есть access_token, который можно использовать в своем приложении для взаимодействия с VK API.

Необходимо заменить пустую строку переменной tkn на полученный ранее токен доступа API. Этот токен будет использоваться для создания сеанса VK API в приложении Flask и получения данных о новостях из VK.

3.3 Создание Flash - приложения

Текущий этап включает в себя непосредственно само создание Flask приложения и является ключевым, поскольку, Flask обеспечивает основу веб-приложения и позволяет взаимодействовать с пользователем.

Для начала создаем файл app.py, который будет содержать все необходимые настройки и код нашего приложения. Затем импортируем необходимые модули (Рис. 2).

```
from flask import Flask, render_template, request
from flask_paginate import Pagination, get_page_parameter, get_page_args
import vk_api
import pytz
from datetime import datetime, timezone
```

Рисунок 2. Импорт необходимых модулей

Затем инициализируем объект Flask (Рис.3).

```
app = Flask(__name__)
app.config['PAGE_SIZE'] = 10  # Количество записей на странице
```

Рисунок 3. Инициализация Flask и определение записей на странице

3.4 Определение класса News

Для представления новости создаем класс News (Рис. 4). В данном случае, этот класс содержит необходимые атрибуты, такие как текст, дата,

идентификатор группы и другие. Важно, чтобы этот класс соответствовал структуре данных, которую ожидается получить от VK API.

```
class News:
    def __init__(self, text, date, group_id, group_name, group_photo, owner_id, post_id):
        self.text = text
        self.date = date
        self.group_id = group_id
        self.group_name = group_name
        self.group_photo = group_photo
        self.owner_id = owner_id
        self.post_id = post_id
```

Рисунок 4. Основной класс, содержащий данные и описание структуры записи

3.5 Получение новостей из VK API

Определяем функцию get_vk_news, которая будет использоваться для взаимодействия с VK API и получения новостей. Нужно обратить внимание на структуру данных, возвращаемую VK API, и убедиться, что код правильно ее обрабатывает (Рис. 5).

```
def get_vk_news(access_token, group_ids, page, per_page):
    vk_session = vk_api.VkApi(token=access_token)
    vk = vk_session.get_api()

    combined_news = []

    offset = (page - 1) * per_page
    count = per_page * len(group_ids)

    for group_id in group_ids:
        try:
            news_feed = vk.wall.get(owner_id=f'-{group_id}', offset=offset, count=count)

            for post in news_feed['items']:
                group_info = vk.groups.getById(group_id=group_id, fields='name,photo_50')
                post['group_info'] = group_info[0]

                utc_time = datetime.fromtimestamp(post['date'], tz=timezone.utc)
                local_time = utc_time.astimezone(pytz.timezone('Asia/Vladivostok'))

                combined_news.append(News(
                    text=post['text'],
                    date=local_time.strftime('%Y-%m-%d %H:%M:%S'),
                    group_id=group_id,
                    group_name=post['group_info']['name'],
                    group_photo=post['group_info']['photo_50'],
                    owner_id=f'-{group_id}',
                    post_id=post['id']
                ))
        except vk_api.exceptions.ApiError as api_error:
            print(f"Ошибка при получении новостей из {group_id}: {api_error}")

    combined_news = sorted(combined_news, key=lambda x: x.date, reverse=True)

    return combined_news[offset:offset + per_page]
```

Рисунок 5. Функция обработки запроса, возвращающая массив записей

3.6 Работа с пагинацией

На этом этапе создаем функции для работы с пагинацией, которые позволят разбить общий список новостей на страницы и создать объект пагинации для передачи в шаблон (Рис. 6).

```
def get_pagination_parameters():
    page = request.args.get(get_page_parameter(), type=int, default=1)
    per_page = app.config['PAGE_SIZE']
    print(f"page: {page}, per_page: {per_page}")
    return page, per_page
```

Рисунок 6. Функция получения числа записей и страниц на их основе

Эта функция использует объект запроса Flask (request) для получения параметров страницы и количества элементов на странице. Если параметры не переданы, устанавливаются значения по умолчанию (страница 1 и количество элементов на странице, определенное в конфигурации приложения)

```
def create_pagination(page, per_page, total_news, group_ids_input):
    group_ids_input = group_ids_input or ''  # Set a default value if it's None
    print(f"total_news in create_pagination: {total_news}")
    return Pagination(
        page=page,
        per_page=per_page,
        total=total_news,
        css_framework='bootstrap4',
        record_name='news_data',
        group_ids=group_ids_input,
        href=request.path + '?page={0}&group_ids=' + (group_ids_input or '')
    )
```

Рисунок 7. Функция создания пагинации страницы

Функция рисунка 7 - create_pagination принимает параметры страницы, количества элементов на странице, общего числа новостей и входных идентификаторов группы. Она использует библиотеку Flask-Paginate для создания объекта пагинации. Здесь устанавливаем CSS-фреймворк 'bootstrap4', задаем имя переменной для записей в шаблоне (record_name), а также формируем URL-ссылку для переключения между страницами.

3.7 Основной маршрут Flask

В последней части кода определен маршрут (@app.route('/')), который обрабатывает GET-запросы. И функция index, в которой получаем: введенные идентификаторы групп, общее количество новостей, новости с помощью get_vk_news, вычисляем сдвиг для пагинации, создаем объект пагинации с помощью create_pagination и возвращаем результат в шаблон. index.html используется для отображения новостей и объекта пагинации. Это позволяет пользователю взаимодействовать с интерфейсом и просматривать новости по страницам (Рис. 8).

```python
@app.route('/', methods=['GET'])
def index():
    group_ids_input = None
    news_data = None
    pagination = None

    if request.method == 'GET':
        group_ids_input = request.args.get('group_ids')
        page, per_page = get_pagination_parameters()
        group_ids = group_ids_input.split(',') if group_ids_input else []
        offset = (page-1)*per_page
        access_token = tkn
        total_news = get_total_news(access_token, group_ids)
        all_news = get_vk_news(access_token, group_ids, page, per_page)
        paginated_news = all_news
        for news_item in all_news:
            print (news_item.date)
        pagination = create_pagination(page, per_page, total_news, group_ids_input)
        print(offset)

        return render_template('index.html', news_data=paginated_news, pagination=pagination, group_ids_input=group_ids_input)

    return render_template('index.html', group_ids_input=group_ids_input, news_data=news_data, pagination=pagination)
```

Рисунок 8. Маршрут запроса и основная функция формирования страницы на основе шаблона index.html

3.8 Основной HTML-шаблон

Определяем HTML-шаблон, который отвечает за отображение данных на веб-странице. Необходимо убедиться, что шаблон корректно оформлен и находится в подкаталоге templates. Структура файла представлена стандартной html структурой с необходимыми тегами фреймфорка для интеграции кода, включающую в себя head, body и его подэлементы: pagination, form, news_data.

3.9 Запуск приложения

Последний этап – это фактический запуск Flask-приложения. Для этого добавим соответствующий код в конце файла app.py (Рис. 9).

```python
if __name__ == '__main__':
    app.run(debug=True)
```

Рисунок 9. Проверка имени приложения и запуск в «Дебаг»-режиме

Эта конструкция гарантирует, что код, находящийся внутри блока, будет выполнен только тогда, когда этот файл запускается напрямую, а не импортирован как модуль в другом скрипте.

Таким образом, после добавления этого кода Flask-приложение готово к запуску. Чтобы его запустить, необходимо выполнить скрипт. В терминале появится сообщение, указывающее на то, что Flask-приложение успешно запущено. После этого можно открыть веб-браузер и перейти по адресу http://127.0.0.1:5000/

Теперь Flask-приложение должно успешно работать, и можно использовать его для агрегации новостей из сообществ VK.

3.10 Разработка страницы сайта

Главная страница – это первая и единственная страница, которую видит пользователь при посещении сайта (Рис. 10-11). На ней представлена основная форма для вывода новостей из агрегируемых сообществ.

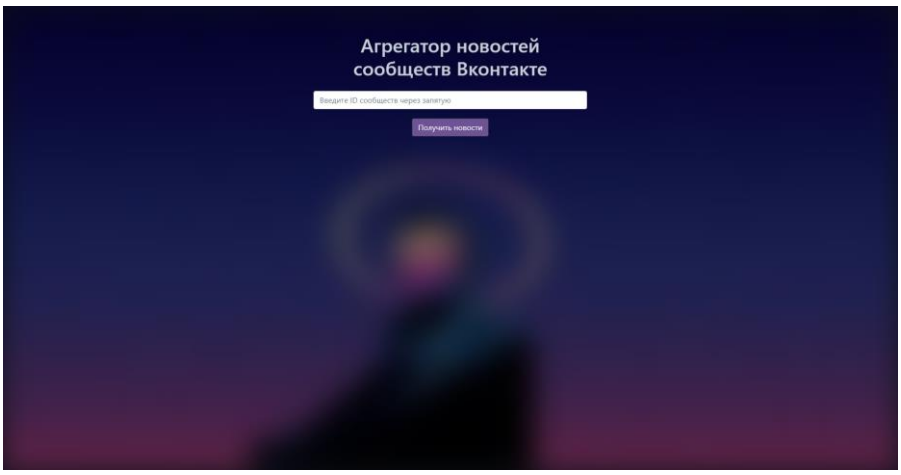

Рисунок 10. Главная страница

```
<h1 class="mt-4 mb-4 home-link floating-text shake" onclick="window.location.href='/'" id="floatingTitle">Агрегатор новостей сообществ Вконтакте</h1>

<form action="/" method="get">
    <div class="form-group">
        <input type="text" class="form-control" id="group_ids" name="group_ids" placeholder="Введите ID сообществ через запятую">
    </div>
    <div class="btn-container">
        <button type="submit" class="btn btn-primary">Получить новости</button>
    </div>
</form>
```

Рисунок 11. Код формы

После отправки запроса с указанными группами на сервер, страница заполняется интерактивными записями этих групп. Для перехода на интересующую запись – можно нажать на нее (Рис. 12-13).

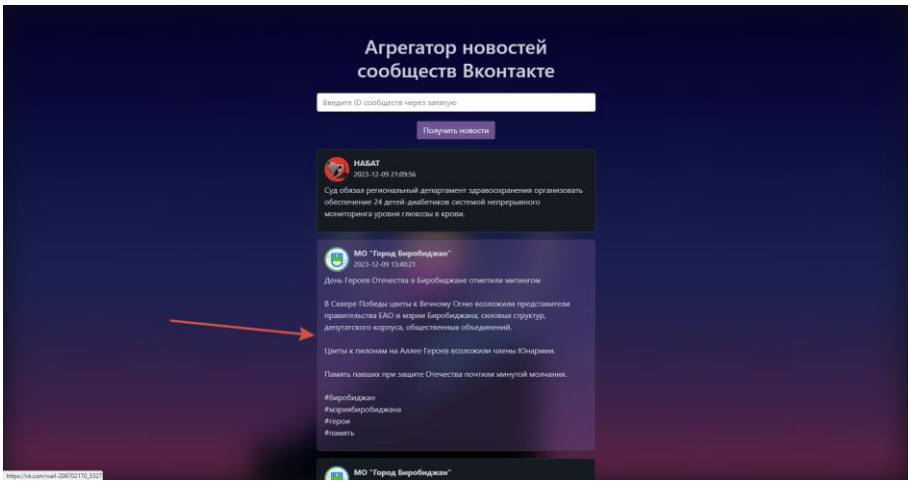

Рисунок 12. Главная страница с отображенными записями

```
{% if news_data %}
<ul>
    {% for news_item in news_data %}
        <a class="block_ev" href="https://vk.com/wall{{ news_item['owner_id'] }}_{{ news_item['post_id'] }}" target="_blank" style="text-decoration: none; color: inherit;">
            <li class="d-flex align-items-start block_ev">
                <div class="vk-icon">
                    <img class="author-icon" src="{{ news_item['group_photo'] }}" alt="Group Logo" width="50" height="50">
                </div>
                <div>
                    <p class="au_name"><b>{{ news_item['group_name'] }}</b></p>
                    <p class="date-info">{{ news_item['date'] }}</p>
                </div>
                <div>
                    <p>{{ news_item['text'] }}</p>
                </div>
            </li>
        </a>
    {% endfor %}
</ul>
{% endif %}
```

Рисунок 13. Код формирования динамического отображения записей главной страницы

Также для удобства пользования кол-во записей на странице было ограничено до 10 штук и добавлена кнопка «Наверх» (Рис.14-15)

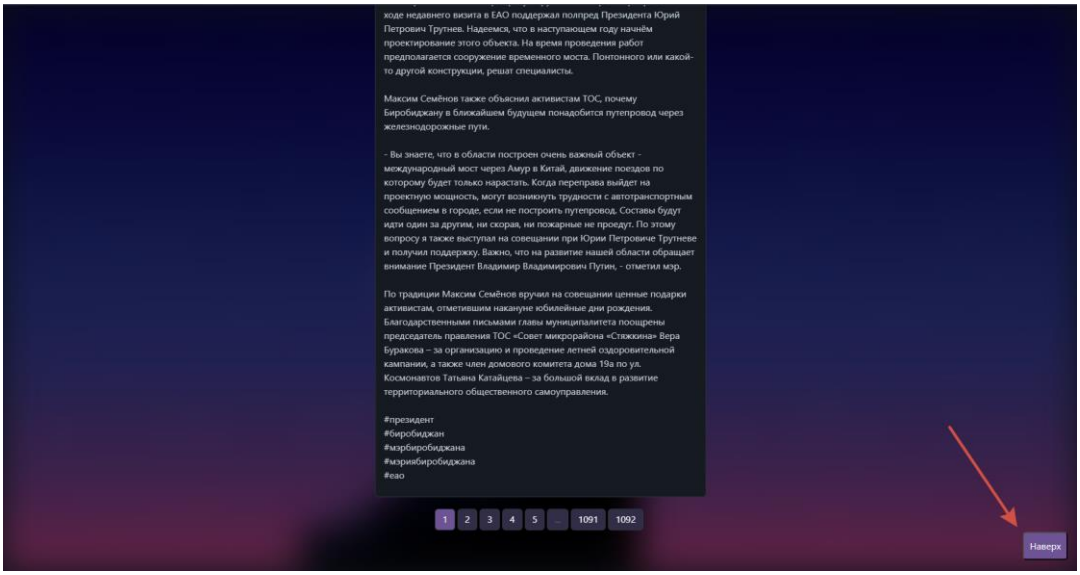

Рисунок 14. Пагинация в нижней части страницы и возврата к верху страницы

```
            {% if pagination %}
                <nav aria-label="Page navigation">
                    <ul class="pagination justify-content-center">
                        {% for page_num in pagination.pages %}
                            {% if page_num %}
                                <li class="page-item{% if page_num == pagination.page %} active{% endif %}">
                                    <a class="page-link" href="{{ url_for('index', page=page_num, group_ids=group_ids_input) }}">{{ page_num }}</a>
                                </li>
                            {% else %}
                                <li class="page-item disabled">
                                    <span class="page-link">...</span>
                                </li>
                            {% endif %}
                        {% endfor %}
                    </ul>
                </nav>
            {% endif %}

        </div>
    </div>
</div>
<script>
    function scrollToTop() {

    document.addEventListener('scroll', function () {
        var scrollPosition = window.scrollY;
        var scrollToTopButton = document.querySelector('.scroll-to-top');

        if (scrollPosition > 200) {
            scrollToTopButton.style.display = 'block';
        } else {
            scrollToTopButton.style.display = 'none';
        }
    });
</script>
<button class="scroll-to-top" onclick="scrollToTop()">Наверх</button>
```

Рисунок 15. Код пагинации и кнопки возврата

3.11 Заключение

Разработка системы включала в себя несколько этапов. На начальном этапе был проведен анализ существующих аналогов, таких как Hootsuite, Flipboard и Feedly. Это позволило определить сильные и слабые стороны существующих решений и учесть их при разработке собственного приложения. В ходе практической работы было создано веб-приложение на основе Flask, обеспечивающее взаимодействие с VK API для получения новостей из выбранных сообществ. Интерфейс пользователя был разработан с учетом удобства и эстетического восприятия, используя HTML, CSS и Bootstrap.

Система позволяет пользователям вводить идентификаторы интересующих сообществ и получать новости в удобном формате. Реализована поддержка пагинации, что обеспечивает удобство работы с большим объемом информации. Также была проведена работа по оптимизации и тестированию приложения, что позволило добиться стабильной и быстрой работы системы.

Выводы

В ходе работы была успешно разработана система агрегации новостей сообществ ВКонтакте. Достигнуты поставленные цели и решены основные задачи. Разработанное приложение предоставляет пользователям удобный инструмент для получения актуальных новостей из различных сообществ социальной сети ВКонтакте. В дальнейшем планируется расширение функционала системы, включая интеграцию с другими социальными платформами и улучшение пользовательского интерфейса.

Библиографический список

1. Moucachen J. Developing a News Aggregation and Validation System // University of Twente. 2017. С. 1-100.
2. Кирьянов, Д.А. Исследование методов построения систем агрегации контента // Программные системы и вычислительные методы. 2022. № 1.
3. Kamble A., Gagan A., Wagh P., Aziz U., Rajput, M. R. Flask Web Framework Based News Summarizer: Web Application // Department of Computer Science and Engineering, P.E.S. College of Engineering, Dr. Babasaheb Ambedkar Technological University, Lonere, Raigad (MH), India. 2022. Т. 5. № 6. С. 953-957.
4. Rai R. K., Singh I., Mudia A., Bisht, K. News Aggregator Web Application Using Django // International Journal of Creative Research Thoughts (IJCRT). 2021. Т. 9. № 7. С. 871-872.
5. Hanji G., Menon A.B., Patil R. A. Nandeni A., Harnessing B. Flask for web scraping and sentiment analysis: a comprehensive application for news and e-commerce reviews // International journal of computer applications, Dept of E & C E, PDACEK, Kalaburagi. 2023. № 16. С. 49-53.