

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего
образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

Кафедра

вычислительной техники

«Вычислительная техника»

Специальность: 09.05.01 «Применение и эксплуатация
автоматизированных систем специального
назначения»
Специализация: «Эксплуатация вычислительных машин,
комплексов, систем и сетей специального
назначения»

ДИПЛОМНЫЙ ПРОЕКТ

на тему

ОПРЕДЕЛЕНИЕ РЕЙТИНГА ТОВАРОВ НА ОСНОВЕ
НЕЙРОСЕТЕВОГО АНАЛИЗА КОММЕНТАРИЕВ В VK

Студент Илья Филиппов 02.06.25 Филиппов Илья Александрович
(подпись, дата) (ФИО полностью)
Руководитель В.Ю. Егоров 02.06.25 Егоров В.Ю.
(подпись, дата) (фамилия, инициалы)
Нормоконтролёр А.С. Бычков 02.06.25 Бычков А.С.
(подпись, дата) (фамилия, инициалы)
Рецензент Доцент Пенз ГТУ Тетерский С.В. 02.06.25 Тетерский С.В.
(должность, место работы) (подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от 28.05.2025 №16)

Заведующий кафедрой Митрохин М.А.
(подпись) (фамилия, инициалы)

Работа защищена с отметкой отлично
(протокол заседания ГЭК от 19.06.2025г. № 5)

Секретарь ГЭК

Е.И. Гурин
(подпись)

Гурин Е.И.
(фамилия, инициалы)

Пенза 2025

Федеральное государственное бюджетное образовательное учреждение
высшего образования

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
Митрохин М.А.
«06» 05 2025 г.

**ЗАДАНИЕ
НА ДИПЛОМНЫЙ ПРОЕКТ**

1. Студент Филиппов Илья Александрович гр. 20ВОЭ1 факультета вычислительной техники.

2. Тема работы: «Определение рейтинга товаров на основе нейросетевого анализа комментариев в vk».

Тема утверждена приказом ПГУ № 601 от 06.05.2025 г.

3. Руководитель работы к.т.н., доцент каф. ВТ «ПГУ» Егоров В.Ю.

4. Задание на дипломную работу (назначение разработки, исходные данные и т.п.).

Разработать клиент – серверное приложение для визуализацию и анализ текстовых данных из открытых источников ВКонтакте в интерактивном пользовательском интерфейсе. Система должна обеспечить ввод ссылки на группу ВКонтакте с валидацией и автоматическим извлечением домена сообщества; осуществлять асинхронную отправку запроса на серверную часть системы с использованием REST API; проводить обработку и отображение данных в виде вкладок, содержащих таблицы постов и комментариев с возможностью раскрытия деталей по каждому элементу; организовывать просмотр расширенной информации по посту во всплывающем окне с возможностью перехода по прямой ссылке на оригинал; обеспечивать обработку исключений и визуальное оповещение пользователя о текущем состоянии (анимация статуса, ошибки соединения, отсутствие данных и т.д.); выполнять нейросетевой анализ всего выгруженного контента, с последующей его оценкой на основе отзывов.

5. Перечень подлежащих разработке вопросов.

Требуется провести анализ предметной области и определить требования к функциональной части, спроектировать архитектуру клиент – серверного приложения, определить стек технологий, необходимых для разработки, разработать диаграммы вариантов использования, последовательностей, компонентов, реализовать пользовательский интерфейс, визуализацию и обработку медеоанных, с анализом и оценкой при помощи нейронной сети.

6. Календарный график выполнения работы.

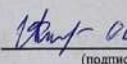
№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области	10	06.05.2025 – 08.05.2025	
2	Проектирование клиентского приложения	10	09.05.2025 – 11.05.2025	
3	Разработка клиентского приложения	30	12.05.2025 – 19.05.2025	
4	Отладка и тестирование	10	20.05.2025 – 22.05.2025	
5	Разработка экономической части	5	23.05.2025	
6	Разработка раздела БЖД	5	24.05.2025	
7	Оформление графической части	10	25.05.2025 – 27.05.2025	
8	Оформление выпускной работы	20	28.05.2025 – 30.05.2025	

Дата выдачи задания « 06 » 05 2025 г.

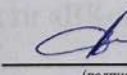
Руководитель дипломного проекта

 06.05.2025 Егоров В.Ю.
(подпись, дата) (фамилия, инициалы)

Задание к исполнению принял студент

 06.05.25 Филиппов И.А.
(подпись, дата) (фамилия, инициалы)

Работу к защите допустить
Декан факультета

 09.06.2025 Фионова Л.Р.
(подпись, дата) (фамилия, инициалы)

Реферат

Пояснительная записка содержит 112 листов, 42 рисунков, 23 таблиц и 29 использованных источников.

СОЦИАЛЬНЫЕ СЕТИ, ПАРСИНГ ДАННЫХ, ВИЗУАЛИЗАЦИЯ ИНФОРМАЦИИ, АНАЛИЗ КОНТЕНТА, ОБРАБОТКА ЕСТЕСТВЕННОГО ЯЗЫКА, REST API, КОММУНИКАЦИЯ КЛИЕНТ–СЕРВЕР, ПОВЕДЕНЧЕСКИЙ АНАЛИЗ

Объект исследования – сервисы нейросетевого анализа активности пользователей на страницах сообществ социальной сети «ВКонтакте».

Цель выпускной квалификационной работы – спроектировать и разработать приложение, обеспечивающее взаимодействие с серверной системой парсинга через REST API, визуализацию и анализ текстовых данных из открытых источников ВКонтакте в интерактивном пользовательском интерфейсе.

В процессе работы необходимо выполнить анализ предметной области, связанной с обработкой и визуализацией пользовательского контента из социальной сети «ВКонтакте»; разработать механизм получения и отображения данных постов и комментариев по заданной группе ВКонтакте; обеспечить интеграцию с серверной частью посредством REST API для запуска парсинга и получения результатов анализа; внедрить оценку тональности текста комментариев с отображением результата.

Результатом работы является полнофункциональное клиентское приложение, предназначенное для визуального анализа и взаимодействия с данными сообщества социальной сети «ВКонтакте».

					ПГУ 1.090501.15.001 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата	Определение рейтинга товаров на основе нейросетевого анализа комментариев в vk Пояснительная записка	Лит.	Лист	Листов
Разраб.		Филиппов И.А.		02.06.25				
Провер.		Егоров В.Ю.		02.06.25			4	112
Н. Контр.		Бычков А.С.		02.06.25		ФВТ, каф. ВТ гр. 20ВОЭ1		
Реценз.		Петерский С.В.		02.06.25				
Утверд.		Митрохин М.А.		02.06.25				

Содержание

Перечень сокращений и обозначений.....	8
Введение.....	9
1 Анализ предметной области и постановка задачи на разработку программных средств автоматизации процесса оценивания товара по тональности отзывов ..	11
1.1 Анализ предметной области аналитики социальных сетей	11
1.2 Анализ программного обеспечения аналогичного назначения.....	12
1.3 Постановка задачи на разработку программных средств	17
2 Техническое задание.....	20
2.1 Назначение разработки	20
2.2 Анализ функциональных требований к программным средствам.....	20
2.3 Анализ нефункциональных требований к программным средствам.....	21
2.4 Требования к программной документации.....	22
2.5 Стадии и этапы разработки	22
2.6 Порядок контроля и приемки.....	23
3 Выбор инструментов разработки	24
3.1 Обоснование выбора языка программирования.....	24
3.2 Используемые библиотеки и зависимости	24
4 Проектирование системы	27
4.1 Общая архитектура системы	27
4.2 Реализация клиентской части приложения.....	28
4.2.1 Общая архитектура клиентской части	29
4.2.2 Пользовательский интерфейс	30
4.2.3 Логика работы с сервером и многопоточность	31
4.2.4 Интерактивная визуализация данных	32
4.2.5 Механизм сохранения данных	33
4.3 Реализация нейросетевого обучения модели	33
4.4 Реализация серверной части приложения.....	35
4.4.1 Архитектура и взаимодействие компонентов	35
4.4.2 Парсинг и запись данных	37

4.4.3	Обработка и анализ комментариев.....	38
4.4.4	Защита и устойчивость	39
4.4.5	Интеграция и масштабируемость	39
5	Проектирование базы данных.....	41
5.1	Разработка структур данных	41
5.2	Концептуальное проектирование	41
5.3	Логическое проектирование.....	43
5.4	Физическое проектирование	45
6	Программа и методика испытаний.....	47
6.1	Объект испытаний.....	47
6.2	Цель испытаний	47
6.3	Требования к программе.....	47
6.4	Требования к программной документации.....	47
6.5	Средства и порядок испытаний.....	48
6.6	Методика испытаний	48
7	Оценка затрат на разработку автоматизированной системы определения рейтинга товаров на основе нейросетевого анализа комментариев	50
7.1	Организация производства	50
7.2	Календарное планирование разработки	53
7.3	Расчет себестоимости программного обеспечения	56
7.4	Оценка эффективности программного обеспечения автоматизированной информационной системы определения рейтинга товаров на основе нейросетевого анализа комментариев в vk	61
7.5	Анализ рисков при разработке и внедрении автоматизированной информационной системы определения рейтинга товаров на основе нейросетевого анализа комментариев в vk	62
8	Общие принципы обеспечения безопасности на рабочем месте оператора информационной-автоматизированной системы салона красоты.....	65
8.1	Актуальность вопросов безопасности человека на рабочем месте.....	65
8.2	Техника безопасности при работе с электроприборами	66

8.3	Влияние ПЭВМ на физиологию человека	67
8.4	Техника безопасности при работе с персональным компьютером	69
	Заключение	71
	Список использованных источников	72
	Приложение А – <i>UML</i> -диаграммы	76
	Приложение Б – Результаты выполнения тестовых примеров	82
	Приложение В – Листинг программы.....	86
	Приложение Г – Презентация	106

Перечень сокращений и обозначений

ПЭВМ – персональная электронно-вычислительная машина

СУБД – система управления базами данных

API (Application Programming Interface) – программный интерфейс приложения

BERT (Bidirectional Encoder Representations from Transformers) - двунаправленная нейронная сеть кодировщик

REST (Representational State Transfer) – передача репрезентативного состояния или передача «самоописываемого» состояния

UML (Unified Modeling Language) – унифицирован язык моделирования.

					ПГУ 1.090501.15.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		8

Введение

В современном мире в эпоху роста потребления информации, появилась необходимость проводить анализ потребляемых данных. Особенно это актуально для социальных сетей, которые трансформировались из средств личной коммуникации в мощнейшие площадки общественного мнения, формирования идеологических повесток и влияния на поведение масс [1]. С увеличением объёмов пользовательского контента возникла объективная необходимость в систематическом и автоматизированном анализе публикуемой информации.

Одной из крупнейших и наиболее влиятельных русскоязычных социальных платформ является «ВКонтакте». Здесь ежедневно публикуются сотни тысяч сообщений, содержащих личные мнения, общественные оценки и эмоциональные реакции на события различного масштаба — от локальных новостей до глобальных информационных поводов. Эти цифровые следы представляют собой ценный материал для анализа настроений, выявления общественных трендов и моделирования поведения аудитории [2].

Анализ сообществ позволяет выявить эмоциональные настроения людей, показать уровень поддержки или критики по различным темам, событиям. Такая информация находит практическое применение в самых разных сферах: от маркетинга и политики до социологии и управления репутацией. Специалисты могут, опираясь на результат обработки, принимать более обоснованные решения, адаптировать коммуникационные стратегии и прогнозировать развитие ситуации [3].

Целью данного проекта является автоматизация процесса сбора и оценки данных, выкладываемых в социальной сети «ВКонтакте». Это позволит не свести к минимуму затраты времени и облегчить работу пользователям, получая на выходе полный анализ, состоящий из оценочных суждений к публикациям и определения тональности к комментариям. Приложение должно представлять удобный графический интерфейс, интуитивно понятного для пользователя, с полным набором функциональных инструментов, включающих извлечение

					ПГУ 1.090501.15.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		9

публикаций, классификацию оценочных суждений и определение тональности комментариев. Таким образом, приложение будет служить надёжным инструментом в руках исследователей, маркетологов, политологов и других специалистов, работающих с большими объёмами социально значимой информации.

1 Анализ предметной области и постановка задачи на разработку программных средств автоматизации процесса оценивания товара по тональности отзывов

1.1 Анализ предметной области аналитики социальных сетей

Каждый день смотря на то, как интернет становится основой нашей жизни, наблюдая за его развитием, становится понятно, что социальные сети стали основным источником получения какой-либо информации. Такие платформы, как ВКонтакте, Telegram и другие, ежедневно пропускают через себя огромные объемы данных. Этот массив комментариев, мнений, обсуждений является ценным ресурсом для определения сентимента и восприятия вашего контента аудиторией [4]. Это помогает подобрать нужную стратегию для работы с информационным потребителем.

Современные системы аналитики представляют из себя автоматизированную систему сбора, обработки и анализа активности пользователей. Самым подходящим способом по анализу мнений людей является определение тональности комментариев, так как именно они дают полное представление об интересующем объекте [5].

Применение подобных средств анализа наиболее актуально в маркетинге, политике, репутационном менеджменте и социологических исследованиях. Пользователь сразу получает результат при помощи графического интерфейса в удобном для него формате.

Тем не менее, готовые платформы для анализа социальных сетей зачастую имеют высокую стоимость, ограниченный функционал настройки под конкретные задачи или вообще разблокирует возможности при оформлении подписки. В таких условиях разработка собственных решений, опирающихся на открытые интерфейсы и современные инструменты анализа текста, становится наиболее востребованной задачей для исследователей и разработчиков [6].

1.2 Анализ программного обеспечения аналогичного назначения

Проанализировав рынок, можно выделить несколько популярных сервисов, которые помогают анализировать отзывы пользователей, вовлеченность аудитории и репутацию брендов во ВКонтакте. Общие сведения о рассматриваемых аналогах приведены в таблице 2.

Таблица 1.1 – Общие сведения о программном обеспечении аналогичного назначения

Название сервиса	Общие сведения
VK.Barkov	Бесплатный инструмент для анализа подписчиков и вовлеченности аудитории ВКонтакте.
Segmento Target	Сервис для анализа целевой аудитории и эффективности рекламных кампаний.
Pepper.Ninja	Инструмент для мониторинга брендов в соцсетях, отслеживания упоминаний и анализа отзывов.
LiveDune	Платформа для аналитики социальных сетей, сбора данных о вовлеченности и популярности контента.
JagaJam	Профессиональный инструмент для анализа отзывов, комментариев и вовлеченности пользователей.
IQBuzz	Платформа для мониторинга соцсетей и СМИ, отслеживания упоминаний бренда и оценки тональности отзывов.

Приведенные аналоги предоставляют различный функционал с различными средствами развертывания. Анализ данных приложений представляет собой выявление сильных и слабых сторон, а также общий вывод по использованию данных приложений.

Интерфейс для работы с веб-сервисом «VK.Barkov» представлен ниже на рисунке 1.1.

К основным преимуществам данного решения можно отнести следующие:

- простота использования;
- бесплатный доступ к основным функциям;
- анализ активности подписчиков.

К недостаткам приложения можно отнести следующие:

- нет детального анализа отзывов;

— ограниченные возможности по изучению тональности.

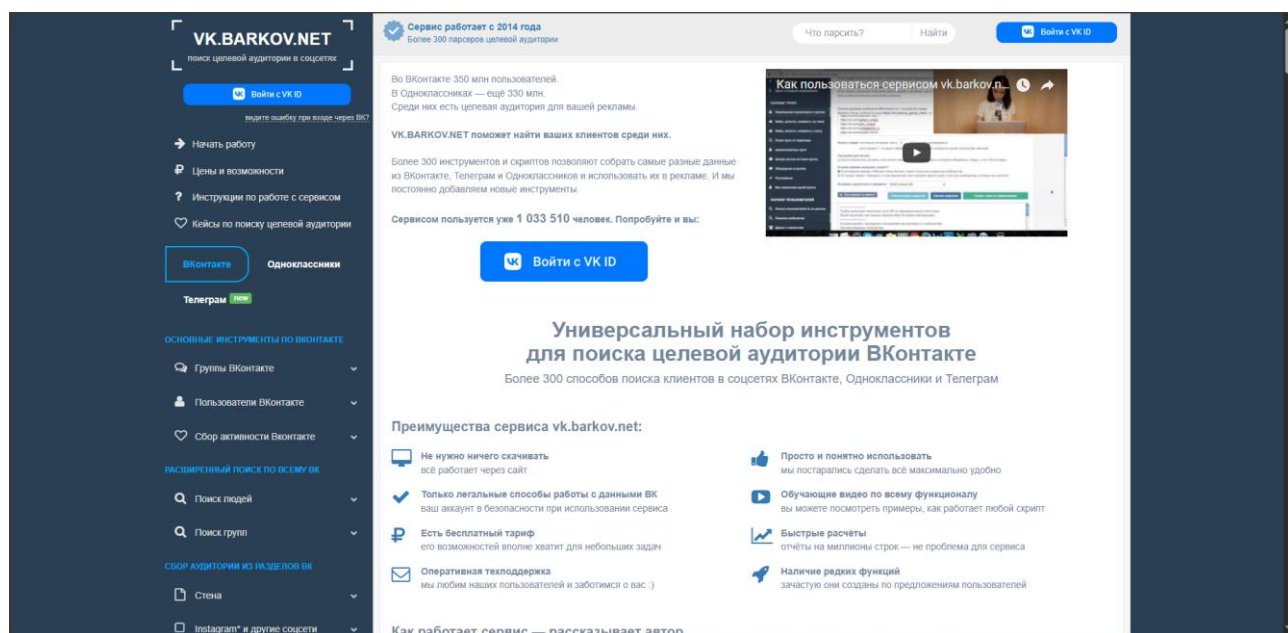


Рисунок 1.1 – Графический интерфейс веб-сервиса «VK.Barkov»

Инструмент Segmento Target для анализа подписчиков и вовлеченности аудитории ВКонтакте. Графический интерфейс данного приложения приведен на рисунке 1.2.

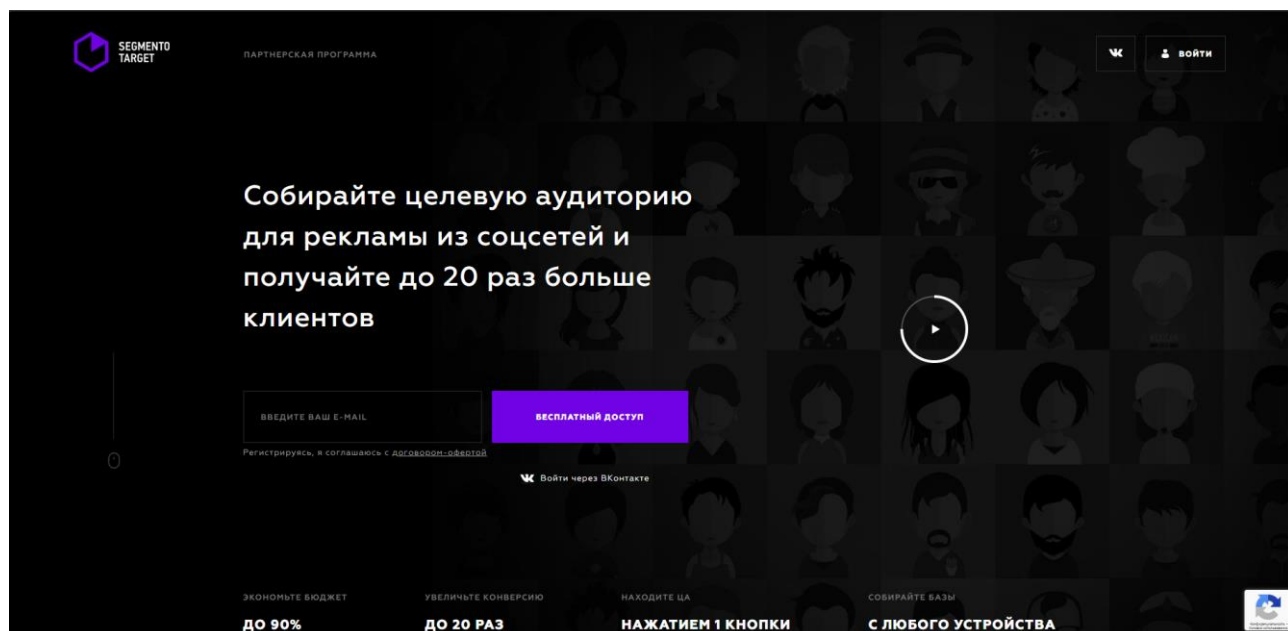


Рисунок 1.2 – Графический интерфейс веб-сервиса «Segmento Target»

К основным преимуществам данного решения можно отнести следующие:

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090501.15.001 ПЗ

Лист

13

- подробный анализ аудитории к;
- глубокий анализ вовлеченности;
- помогает оптимизировать рекламные кампании.

К недостаткам приложения можно отнести следующие:

- высокая стоимость;
- не предназначен для анализа тональности отзывов.

Сервис «Pepper.Ninja» для мониторинга брендов в соцсетях, отслеживания упоминаний и анализа отзывов. Графический интерфейс инструмента представлен на рисунке 1.3.

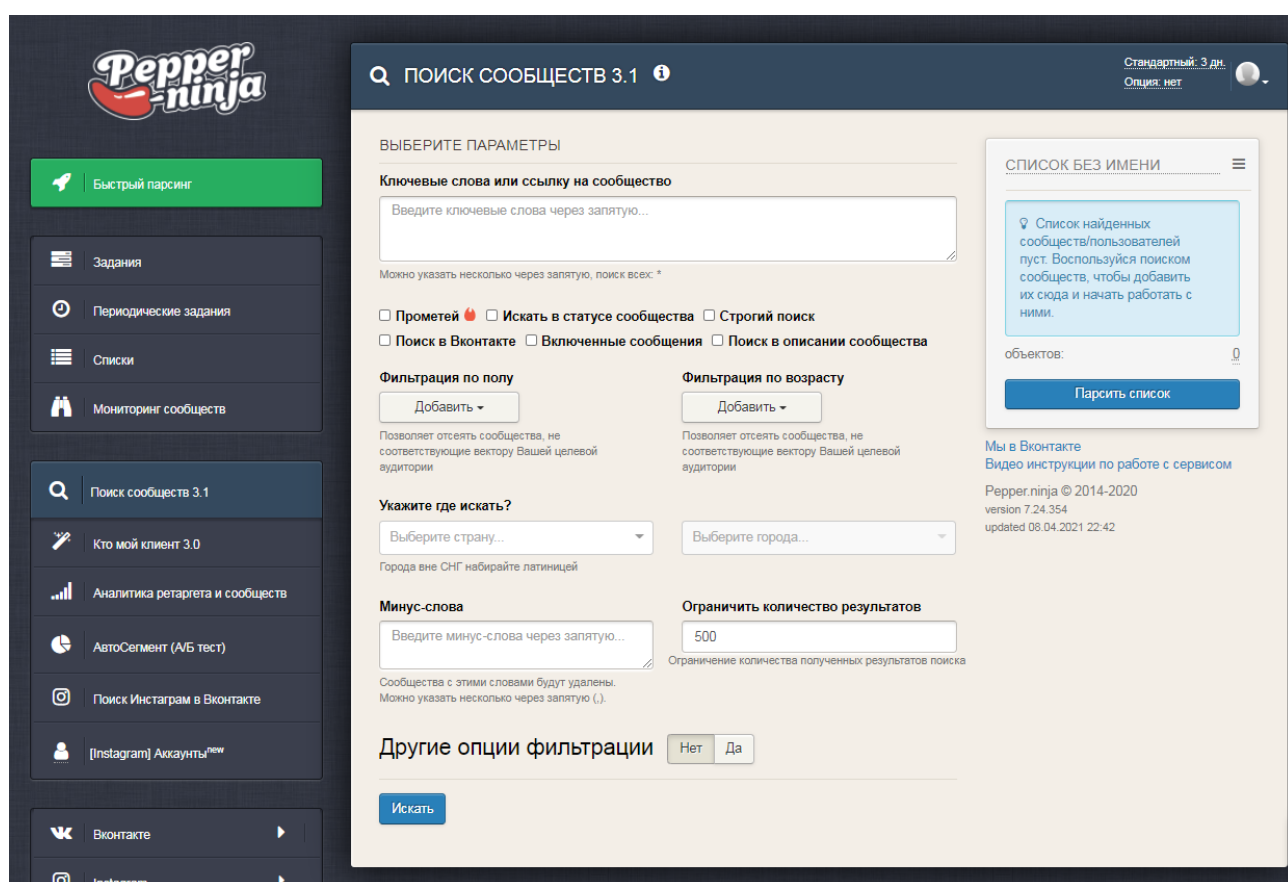


Рисунок 1.3 – Графический интерфейс инструмента «Pepper.Ninja»

К основным преимуществам данного решения можно отнести следующие:

- расширенный мониторинг упоминаний;
- анализ тональности комментариев;
- поддержка нескольких соцсетей.

К недостаткам приложения можно отнести следующие:

- подписка может быть дорогой;
- сложный интерфейс для новичков.

Платформа «LiveDune» для аналитики социальных сетей, сбора данных о вовлеченности и популярности контента. Графический интерфейс данного приложения представлен на рисунке 1.4.

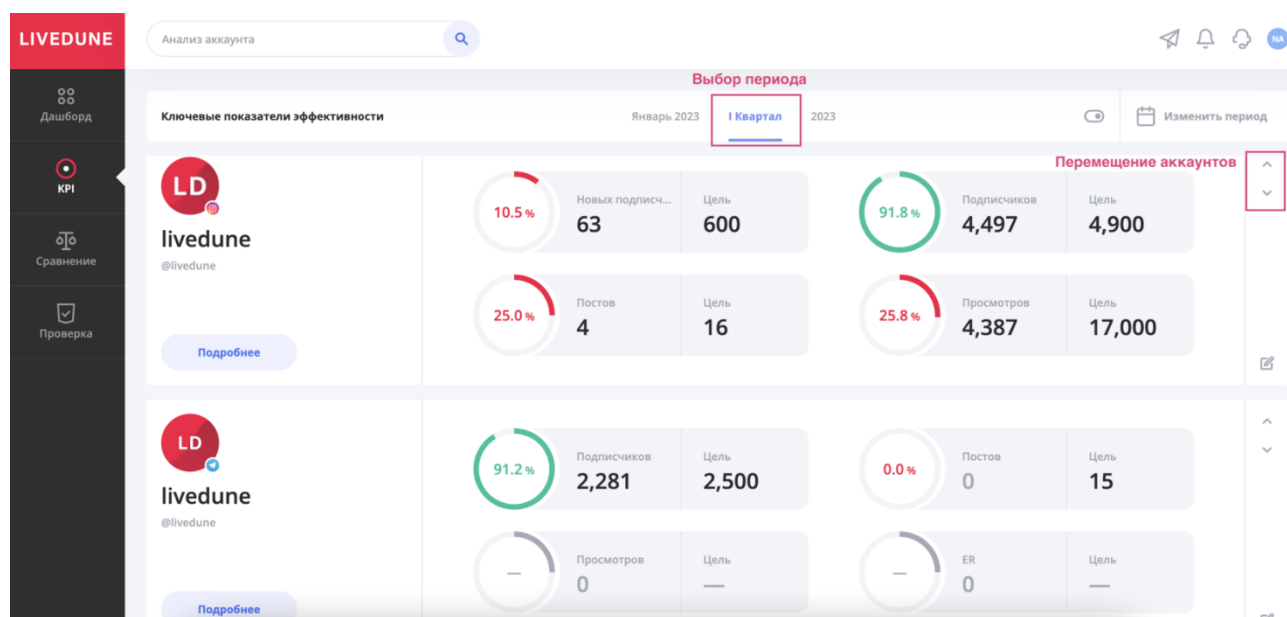


Рисунок 1.4 – Графический интерфейс веб-сервиса «LiveDune»

К основным преимуществам данного решения можно отнести следующие:

- подробная аналитика вовлеченности;
- автоматическое создание отчётов;
- инструменты для управления контентом.

К недостаткам приложения можно отнести следующие:

- нет глубокого анализа тональности;
- большая часть функций доступна только по подписке.

Профессиональный инструмент «JagaJam» для анализа отзывов, комментариев и вовлечённости пользователей. Графический интерфейс данного приложения представлен на рисунке 1.5.

К основным преимуществам данного решения можно отнести следующие:

- глубокий анализ тональности;

- удобные отчёты с визуализацией;
- поддержка анализа вовлеченности.

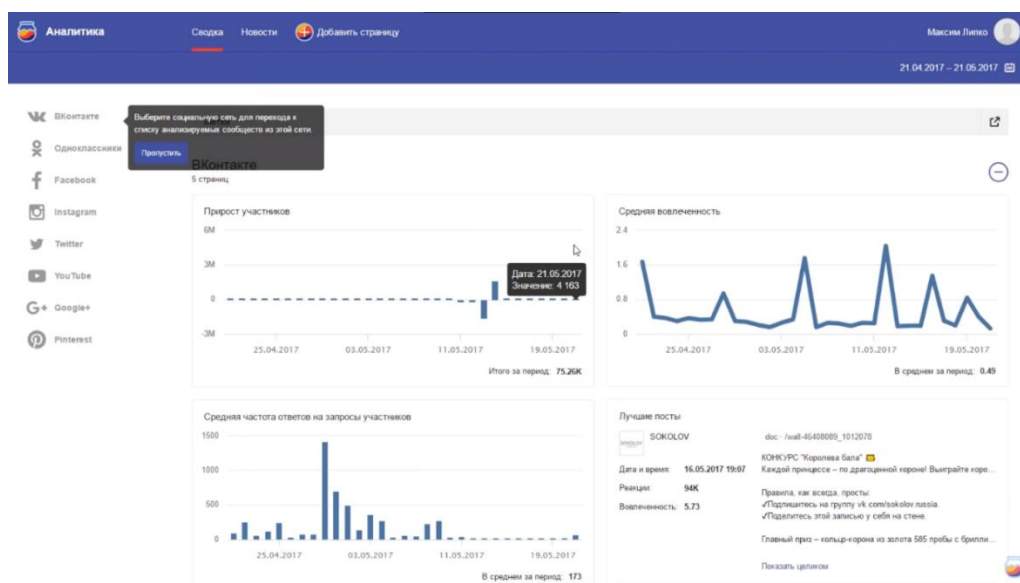


Рисунок 1.5 – Графический интерфейс веб-сервиса «JagaJam»

К недостаткам приложения можно отнести следующие:

- высокая цена;
- требует обучения для эффективного использования.

Платформа «IQBuzz» для мониторинга соцсетей и СМИ, отслеживания упоминаний бренда и оценки тональности отзывов. Графический интерфейс данного приложения представлен на рисунке 1.6.



Рисунок 1.6 – Графический интерфейс веб-сервиса «IQBuzz»

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090501.15.001 ПЗ

К основным преимуществам данного решения можно отнести следующие:

- мониторинг в реальном времени;
- анализ тональности комментариев;
- выявление кризисных ситуаций.

К недостаткам приложения можно отнести следующие:

- сложный интерфейс;
- дорогая подписка.

Для проведения сравнительного анализа приложений аналогов был рассмотрен набор критериев, на основе которых производится сравнение. Результаты сравнения по заданным критериям приведены в таблице 1.2.

Таблица 1.2 – Результаты сравнения разрабатываемых программных средств с аналогами

Сервис	Анализ тональности	Мониторинг упоминаний	Анализ вовлеченности	Автоматизация контента	Анализ ЦА	Цена (высокая)
VK.Barkov	Нет	Нет	Да	Нет	Нет	Нет
Segmento Target	Нет	Нет	Да	Нет	Да	Да
Pepper.Ninja	Да	Да	Нет	Нет	Нет	Да
LiveDune	Нет	Нет	Да	Да	Нет	Да
JagaJam	Да	Да	Да	Нет	Нет	Да
IQBuzz	Да	Да	Нет	Нет	Нет	Да

На основе результатов сравнения можно сделать вывод, что программные средства автоматизации анализа трафика соц. сетей в полной мере выполняют свою работу в рамках заявленных функций. Разрабатываемое приложение будет выполнять анализ тональности, иметь низкую цену, автоматизацию контента.

1.3 Постановка задачи на разработку программных средств

Основной задачей на разработку программных средств автоматизации аналитики социальной сети «ВКонтакте» является создание приложения,

позволяющего выполнять сбор, обработку и оценивание информационного потока.

Приложение должно обеспечивать выгрузку данных и взаимодействовать с внедренной моделью машинного обучения для определения эмоциональной окраски комментариев

Приложение должно обеспечивать подключение к онлайн-платформе «ВКонтакте» для получения данных. Это включает выполнение запросов и извлечение текстов комментариев. Сам процесс получения данных должен быть автоматизированным, поддерживать работу с большими объемами информации и выводить состояние процесса обработки информации пользователю. Также к особенностям будет относиться работа приложения в формате клиент-сервер с использованием http запросов для взаимодействия между сторонами.

Так как анализируется большой объем данных, приложение должно быть оптимизировано для работы с большими массивами текстов. Важно, чтобы процесс обработки был быстрым, не перегружал вычислительные ресурсы и мог работать в потоковом режиме. Это позволит анализировать комментарии сразу после их публикации и оперативно обновлять статистику.

После анализа тональности приложение должно сохранять полученные результаты. Для этого требуется СУБД, которая позволит фиксировать данные о каждом комментарии, связанной с ним информации, такой как идентификатор поста и дата публикации, а также его эмоциональной окраске. Также должна быть возможность экспорта данных в удобном формате для последующего использования в иных средах разработки или анализа.

На основе выделенных признаков принадлежности инструментов аналитики социальных сетей, можно сформировать следующие разделы программы:

- главная страница;
- несколько вкладок для разделения проанализированной информации;
- выводы постов.

Таким образом, на основе сформированных требований будут произведены проектирование и разработка программных средств автоматизации аналитики социальной сети «ВКонтакте».

					ПГУ 1.090501.15.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		19

2 Техническое задание

2.1 Назначение разработки

Разрабатываемое программное обеспечение представляет собой клиент – серверное приложение [7], предназначенного для сбора данных страницы сообщества «ВКонтакте», проведения нейросетевого анализа комментариев общества, вывода оценки товара из проведенного изучения, вывода структурированной совокупности информации и предоставление возможности взаимодействия пользователя данного приложения с полученной информацией.

2.2 Анализ функциональных требований к программным средствам

Исходя из предметной области и задач данной разработки, были сформирован ряд функциональных требований относительно данного приложения. Основной целью является реализация клиент - серверного приложения, позволяющего производить парсинг открытых данных сообщества социальной сети «ВКонтакте», представление и взаимодействие с полученными данными в структурированной табличной форме.

Пользовательский сценарий взаимодействия представлен на диаграмме вариантов использования [8]:

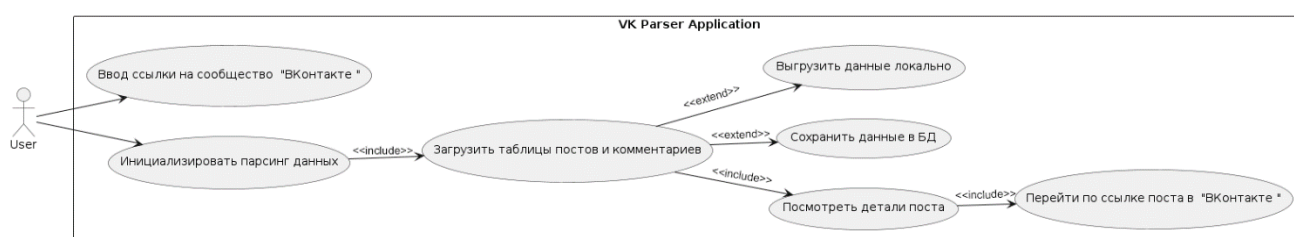


Рисунок 2.1 – Диаграмма вариантов использования приложения

На диаграмме отражены следующие основные действия пользователя:

1. Ввод ссылки на сообщество «ВКонтакте» – пользователь предоставляет URL-адрес интересующего сообщества, по которому будет осуществлён сбор информации.

2. Инициализация парсинга данных – по нажатию на соответствующую кнопку инициируется процесс отправки запроса к серверу, где происходит обработка, сбор и первичная агрегация данных.

3. Загрузка таблиц постов и комментариев – по завершении парсинга происходит отображение результатов в виде интерактивных таблиц с возможностью адаптивной обрезки текста и просмотра всплывающих подсказок.

4. Сохранение данных – реализована возможность выгрузки собранной информации в локальный JSON-файл для последующего анализа либо хранения.

5. Просмотр деталей поста – при двойном клике по строке таблицы открывается модальное окно с подробной информацией по выбранному посту.

6. Переход по ссылке на пост во «ВКонтакте» – предусмотрена возможность перехода на оригинальную публикацию в социальной сети для проверки или взаимодействия с ней.

Таким образом, сформулированные требования обеспечивают реализацию интуитивно понятного и функционально полного интерфейса взаимодействия пользователя с системой сбора и анализа данных «ВКонтакте».

2.3 Анализ нефункциональных требований к программным средствам

Разработка программных средств автоматизации аналитики социальной сети «ВКонтакте», исходящей из тональности отзывов, должна быть выполнена в многофункциональной среде разработки Microsoft Visual Studio Code 1.99.3 на объектно-ориентированном языке программирования Python 3.13.2. Графический интерфейс и остальные фрагменты приложения полностью построены с использованием данного языка. Приложение должно разворачиваться и исполняться на операционных системах семейства Windows версии 10 и выше.

2.4 Требования к программной документации

Разрабатываемое программное средство для извлечения и визуального анализа контента из сообщества «ВКонтакте» должно сопровождаться комплектом программной документации, включающим следующие разделы:

- Техническое задание;
- Описание архитектуры программной системы;
- Описание процесса разработки;
- Методика тестирования;
- Описание развертывания и эксплуатации;
- Инструкция пользователя.

Структура и содержание программной документации должны соответствовать требованиям Единой системы программной документации (ЕСПД) [9].

2.5 Стадии и этапы разработки

Этапы разработки клиент–серверного приложения анализа товаров по тональности комментариев представлены в таблице 1.

Таблица 2.1 – Стадии и этапы разработки

Стадии	Этапы	Объем работы в процентах	Срок выполнения	Исполнитель
Анализ предметной области	Анализ предметной области	10	06.05.2025 – 08.05.2025	Филиппов И.А.
Проектирование системы	Описание архитектуры	10	09.05.2025 – 11.05.2025	Филиппов И.А.
Разработка системы	Разработка системы	20	12.05.2025 – 19.05.2025	Филиппов И.А.
Отладка и Тестирование	Отладка и тестирование	10	20.05.2025 – 22.05.2025	Филиппов И.А.
Разработка экономической части	Оценка затрат на разработку и внедрение программного приложения контроля выполнения задач	5	23.05.2025	Филиппов И.А.

Стадии	Этапы	Объем работы в процентах	Срок выполнения	Исполнитель
Разработка раздела БЖД	Оценка тяжести труда эксплуататора программного приложения	5	24.05.2025	Филиппов И.А.
Оформление графической части	Оформление графической части	10	25.05.2025 – 27.05.2025	Филиппов И.А.
Оформление выпускной работы	Оформление пояснительной записки	30	28.05.2025 – 30.05.2025	Филиппов И.А.

2.6 Порядок контроля и приемки

Для подтверждения соответствия разработанного программного средства заявленным в техническом задании требованиям необходимо провести серию испытаний, охватывающих все ключевые сценарии его использования.

Программное средство считается соответствующим техническому заданию, если при подаче различных тестовых входных данных обеспечивается выполнение всех предусмотренных функций, включая устойчивую работу интерфейса, сохранность данных, точность визуализации и корректную обработку пользовательских действий.

Приемка программного продукта должна производиться в рабочем состоянии системы, в присутствии заказчика, с демонстрацией всех реализованных функций. Обязательным условием является наличие полной программной документации, включая:

- пояснительную записку,
- исходный код программного средства,
- результаты тестовых прогонов.

Факт успешной приемки подтверждается протоколом испытаний и актом приёмки, подписанным обеими сторонами.

3 Выбор инструментов разработки

3.1 Обоснование выбора языка программирования

Для разработки клиентской части проекта был выбран язык программирования Python. Благодаря наличию проверенных инструментов для работы с сетью, обработки данных и реализации многопоточности, а также для создания графических интерфейсов, Python предоставляет все необходимые средства для реализации функционала парсинга, обработки и визуализации данных.

Особое внимание стоит уделить простоте и надежности интеграции с REST API [10]. Встроенные возможности Python для выполнения HTTP-запросов обеспечивают гибкое взаимодействие клиентского приложения с серверной частью, что критично для корректной работы проекта.

Также благодаря фреймворку FastAPI [11], который позволяет написать свое API под серверную часть и поддерживает асинхронное выполнение задач. Это позволяет обрабатывать множество запросов одновременно, не блокируя выполнение других операций.

3.2 Используемые библиотеки и зависимости

Для реализации клиент – серверного приложения нейросетевого анализа тональности товара в сообществах социальной сети «ВКонтакте» было использовано ряд библиотек необходимых для реализации всех функций данного проекта. Выбор каждой из них основан на ключевых требованиях к функционалу системы, включая просмотр постов и комментариев, с последующим анализом.

Для работы с базой данных PostgreSQL применяется библиотека psycopg2, которая предоставляет надежный и эффективный интерфейс для выполнения SQL-запросов, обеспечивая взаимодействие с серверной частью и хранение обработанных данных [12].

Обработка и анализ табличных данных выполняется с помощью pandas — одной из самых мощных и популярных библиотек для анализа данных в Python.

Благодаря ей вы можете удобно преобразовывать, фильтровать и агрегировать данные из социальных сетей для последующей визуализации и анализа [13].

Для создания веб-сервиса и API взаимодействия используется FastAPI, современный фреймворк, известный своей высокой производительностью и удобством разработки REST API. Pydantic применяется для валидации и сериализации данных, обеспечивая корректный обмен информацией между клиентской и серверной частями [14].

Для реализации клиентского интерфейса и визуализации данных выбран customtkinter — современное расширение стандартной библиотеки Tkinter, которое позволяет создавать удобные, адаптивные и стильные графические интерфейсы с минимальными усилиями [15]. Использование также стандартных модулей tkinter и ttk позволяет эффективно работать с базовыми элементами интерфейса и виджетами.

Для сетевого взаимодействия и отправки HTTP-запросов применяется библиотека requests, что обеспечивает гибкий и надежный обмен данными с внешними сервисами и API [16].

Важной частью проекта является применение методов машинного обучения и нейросетевых моделей. Для этого используются PyTorch и связанные с ним модули: torch.nn для построения и обучения нейросетей [17], torch.optim для оптимизации моделей [18], а также transformers от Hugging Face для работы с предобученными моделями BERT. Ключевые компоненты — это BertTokenizer и BertForSequenceClassification, которые обеспечивают токенизацию текста и классификацию, что позволяет выполнять анализ контента постов и комментариев [19].

Для обработки и загрузки данных в нейросети применяется классический инструмент — torch.utils.data.Dataset и DataLoader, обеспечивающие удобное управление большими объемами данных при обучении и инференсе моделей [20].

Параллельная обработка и поддержание отзывчивости интерфейса обеспечиваются с помощью threading — модуля для многопоточности, что

критично при выполнении длительных операций, таких как парсинг данных или обучение моделей [21].

Дополнительно проект включает работу с системными и вспомогательными модулями: os для управления файловой системой, re для обработки регулярных выражений, json для сериализации данных и webbrowser для открытия внешних ссылок, что расширяет возможности взаимодействия пользователя с приложением [22].

Использование прогресс-бара реализовано с помощью tqdm, что улучшает пользовательский опыт во время выполнения долгих задач [23].

Исходя из этого, подбор и сочетание данных библиотек и модулей отражает цель создать мощное, функциональное, масштабируемое и удобное приложение, способное эффективно собирать, анализировать и визуализировать данные из социальной сети с применением современных методов машинного обучения и удобного пользовательского интерфейса.

4 Проектирование системы

4.1 Общая архитектура системы

Система состоит из трёх ключевых компонентов, тесно взаимодействующих между собой и обеспечивающих полноценный цикл анализа тональности публикаций и комментариев из социальной сети Вконтакте:

- Клиентская часть представляет собой настольное приложение на базе customtkinter. Она отвечает за ввод данных пользователя, таких как домен группы ВКонтакте, и визуализацию результатов — постов, комментариев и анализа тональности. Клиент взаимодействует с сервером по HTTP через REST API.

- Серверная часть построена на фреймворке FastAPI и реализует бизнес-логику системы. Она предоставляет интерфейс REST API для приёма запросов на парсинг и анализ данных из ВКонтакте. Сервер обрабатывает запросы, парсит данные через модуль VKParse, сохраняет информацию в реляционной базе данных PostgreSQL и запускает модель тонального анализа, построенную на основе предобученного трансформера RuBERT.

- Модель машинного обучения загружается в память сервера и используется для классификации тональности комментариев в реальном времени. Она обучена на специализированном датасете с тремя классами: положительный, нейтральный и отрицательный. Результаты анализа интегрируются с данными из базы и передаются клиенту для отображения.

База данных PostgreSQL хранит структурированную информацию о постах и комментариях, обеспечивая согласованность данных с помощью ссылочной целостности и индексов.

Взаимодействие компонентов происходит следующим образом:

- Клиент отправляет запрос на сервер с доменом группы ВКонтакте.
- Сервер парсит и сохраняет посты и комментарии в базе данных.
- Затем запускается анализ тональности комментариев с помощью модели.
- Итоговые данные с оценками тональности возвращаются клиенту в JSON-формате.

- Клиент визуализирует результаты пользователю.

Схема взаимодействия основных компонентов системы представлена на рисунке 4.1.

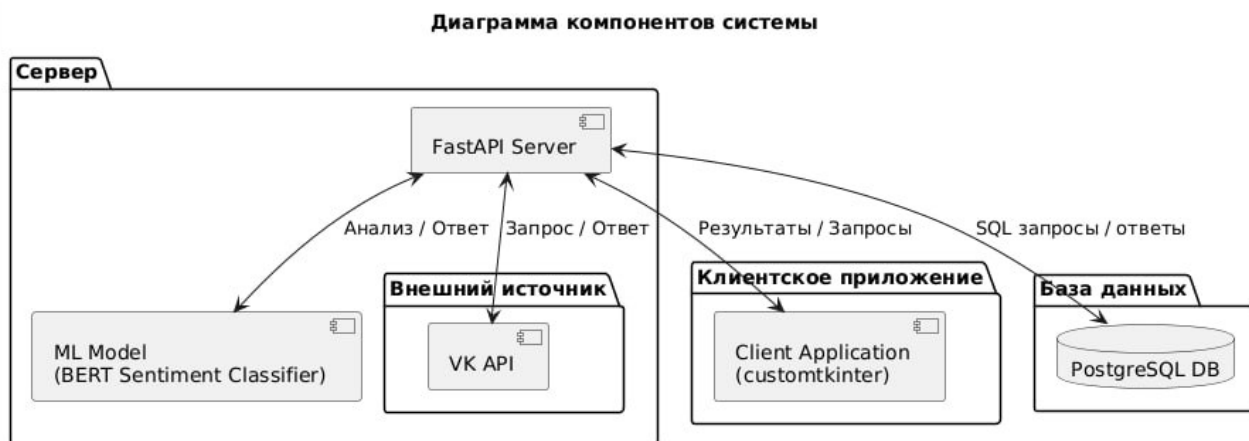


Рисунок 4.1 — Архитектура компонентов системы

4.2 Реализация клиентской части приложения

Клиентская часть программного комплекса представляет собой самостоятельный графический модуль, предназначенный для обеспечения удобного взаимодействия конечного пользователя с функциональностью анализа и извлечения данных из социальной платформы «ВКонтакте». Её основная задача — предоставить визуально понятный и логически структурированный интерфейс, упрощающий процесс получения информации о публикациях и комментариях в выбранных сообществах.

Архитектура интерфейса выстроена с ориентацией на интуитивное восприятие: пользователь может ввести ссылку на интересующую группу, инициировать процесс извлечения данных, отслеживать статус текущих операций, а также просматривать и сохранять полученные таблицы с результатами анализа. Это позволило сделать интерфейс не только эстетичным, но и технологически гибким.

4.2.1 Общая архитектура клиентской части

Клиентская часть организована модульно, что способствует читаемости кода, упрощению отладки и будущему масштабированию. Каждый модуль отвечает за строго определённый функционал:

- Главный исполняемый модуль запускает приложение и инициализирует главное окно.
- Основной графический интерфейс описан в отдельном модуле, содержащем класс с виджетами, обработчиками событий и методами взаимодействия с сервером.
- Визуальное оформление и стили интерфейса выделены в отдельный файл, что позволяет централизованно управлять внешним видом приложения.

Архитектура клиентской части представлена на рисунке 4.2.

Архитектура клиентской части VK Group Parser

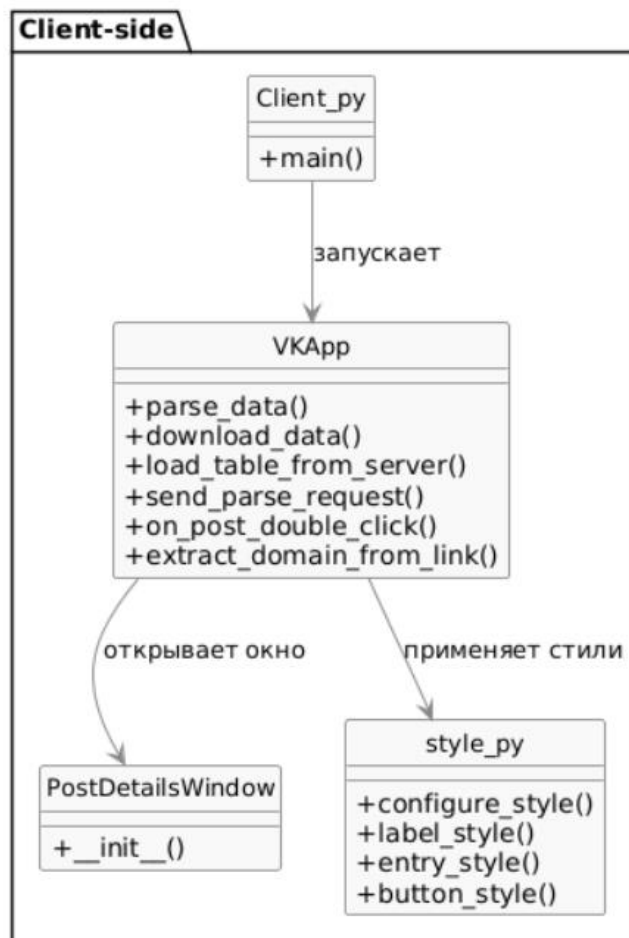


Рисунок 4.2 — Архитектура клиентской части системы

4.2.2 Пользовательский интерфейс

Интерфейс приложения проектировался с учётом простоты использования. Главное окно приложения представлено на рисунке 4.3 включает несколько ключевых компонентов, каждый из которых выполняет отдельную роль в процессе работы:

- Поле ввода ссылки на группу «ВКонтакте» позволяет пользователю быстро указать целевой ресурс.
- Кнопка запуска парсинга инициирует взаимодействие с серверной частью, отправляя введённую ссылку на обработку.
- Статусная строка информирует пользователя о ходе выполнения операций, отображая текущий этап работы.
- Кнопка скачивания становится активной после завершения парсинга и позволяет сохранить данные в форматах JSON и EXCEL.
- Область вкладок служит для отображения результатов: постов и комментариев в виде интерактивных таблиц.

Такой интерфейс обеспечивает полное покрытие пользовательских сценариев — от запуска до анализа и сохранения результата.

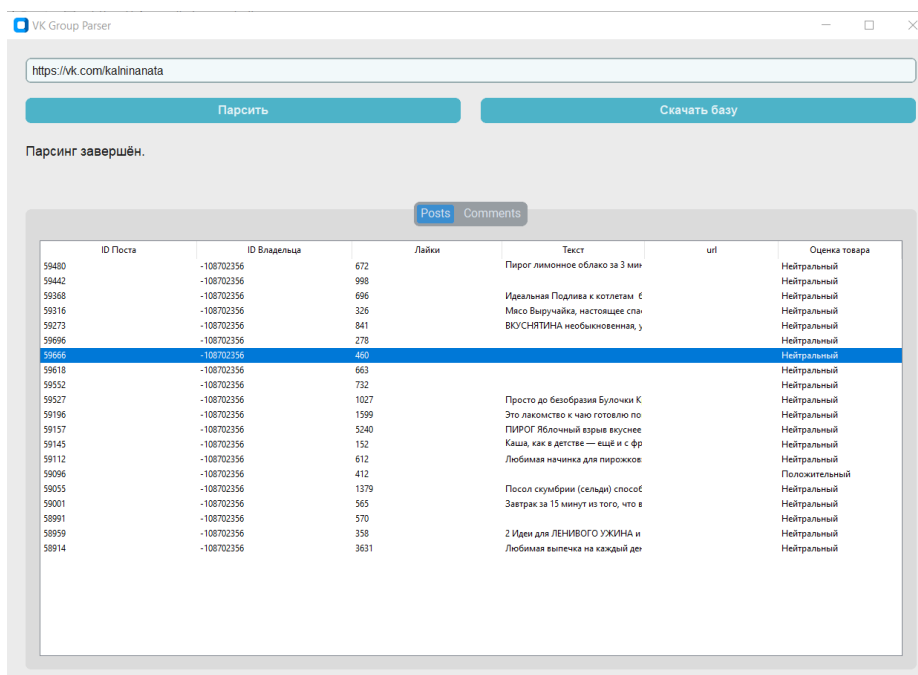


Рисунок 4.3 — Главный экран приложения

4.2.3 Логика работы с сервером и многопоточность

Ключевой задачей клиентской части является асинхронное взаимодействие с серверной частью, где непосредственно происходит парсинг данных. Для этого в приложении реализована система запуска фоновых потоков. Это позволяет избежать «замораживания» интерфейса при длительной обработке, обеспечивая комфортную работу пользователя.

Каждое сетевое обращение к серверу обернуто в отдельный поток. При запуске парсинга клиент отправляет POST-запрос с доменом целевой группы и получает статус выполнения. После успешного завершения клиент запрашивает собранные данные и отображает их в виде таблиц. Весь процесс работы с сервером показан на диаграмме последовательности (рисунок 4.4).

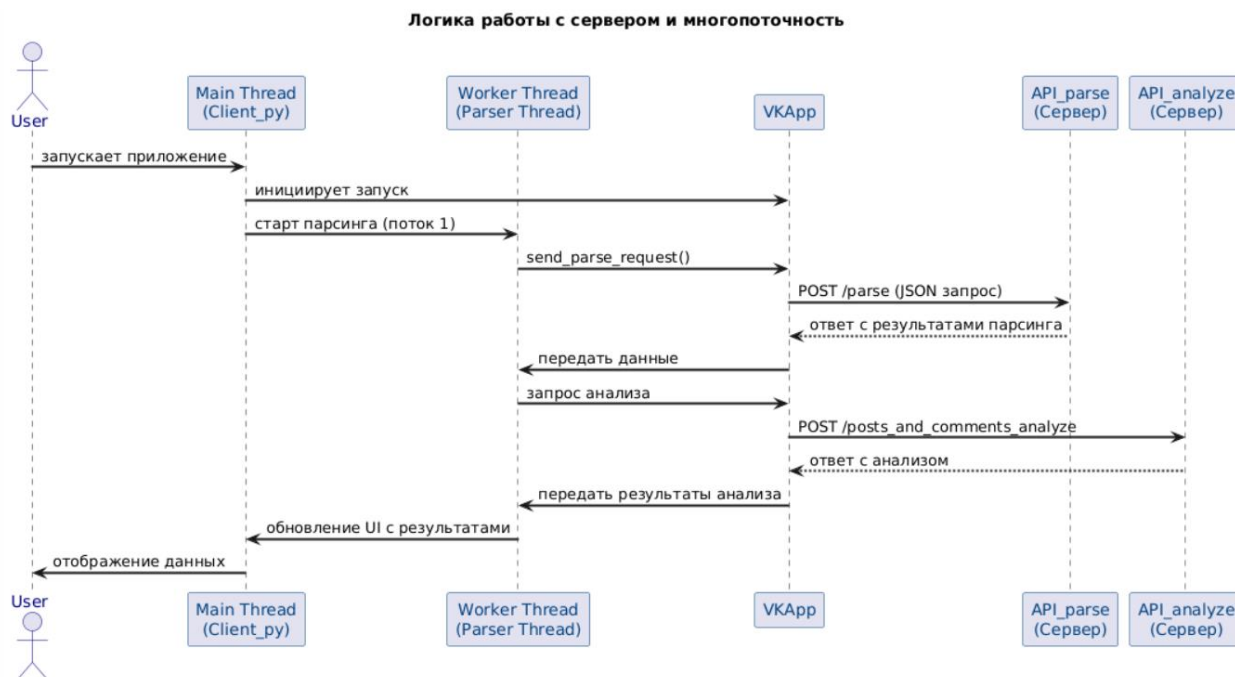


Рисунок 4.4 — Диаграмма последовательности работы клиента - сервера

Если во время выполнения запроса происходит ошибка — пользователь немедленно получает уведомление в интерфейсе. Такой подход обеспечивает отказоустойчивость и повышает доверие пользователя к системе.

4.2.4 Интерактивная визуализация данных

Полученные данные (посты и комментарии) отображаются в виде табличных представлений. Каждая таблица реализована на основе стандартного Treeview, что обеспечивает следующие возможности:

- Простое восприятие информации;
- Поддержка сортировки и фильтрации в будущем;
- Возможность выделения и просмотра деталей.

Для удобства пользователя предусмотрено раскрытие подробной информации о каждом посте (рис. 4.5). При двойном клике по строке таблицы открывается новое окно с подробной информацией, где можно прочитать текст поста, увидеть результат анализа выбранного товара и перейти по прямой ссылке на саму публикацию в социальной сети. Это реализовано с максимальной простотой — переход в браузере осуществляется по нажатию на гиперссылку, оформленную как интерактивный элемент GUI.

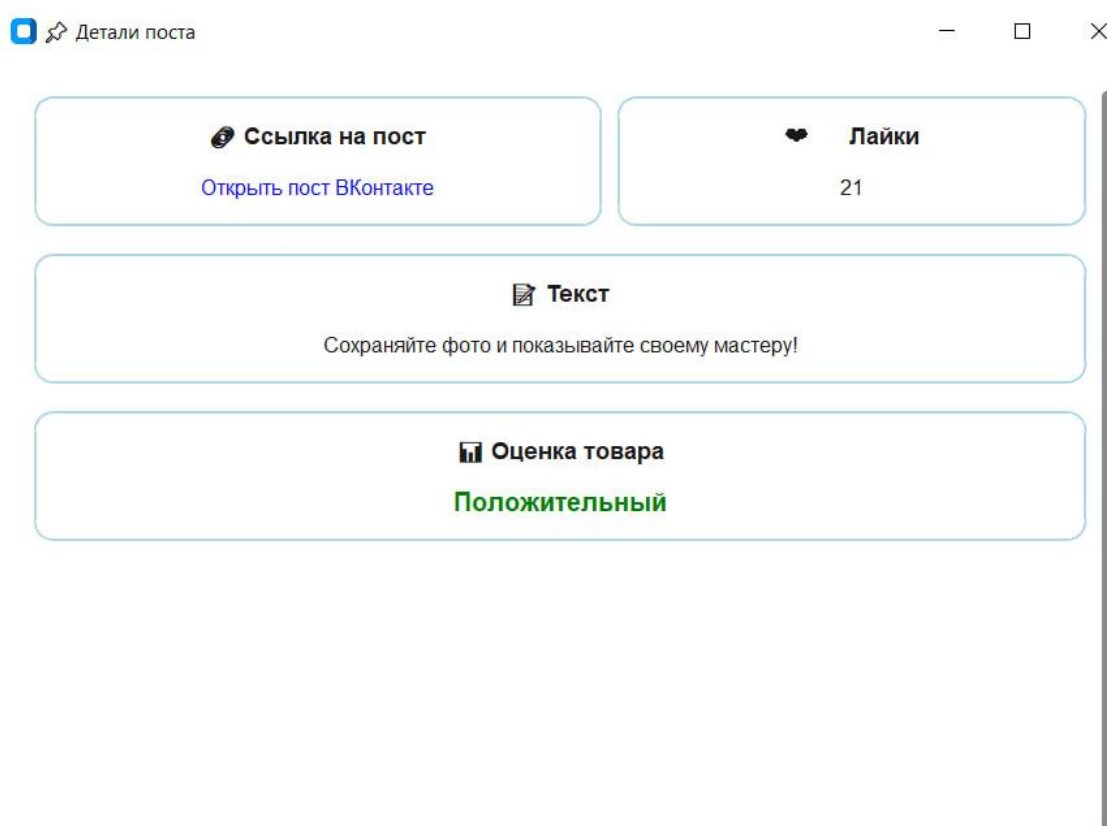


Рисунок 4.5 — Работа механизма раскрытия поста

4.2.5 Механизм сохранения данных

Клиентская часть предоставляет удобную функцию сохранения информации в двух наиболее востребованных форматах:

- JSON — для дальнейшей работы с данными в программном обеспечении.
- Excel (XLSX) — для анализа и обработки в табличных редакторах.

В момент вызова сохранения пользователю предлагается выбрать тип файла и путь его размещения (рис. 4.6). Если пользователь отменяет выбор — сохранение не выполняется. Все потенциальные ошибки, возникающие во время записи на диск, обрабатываются и сопровождаются понятными сообщениями.

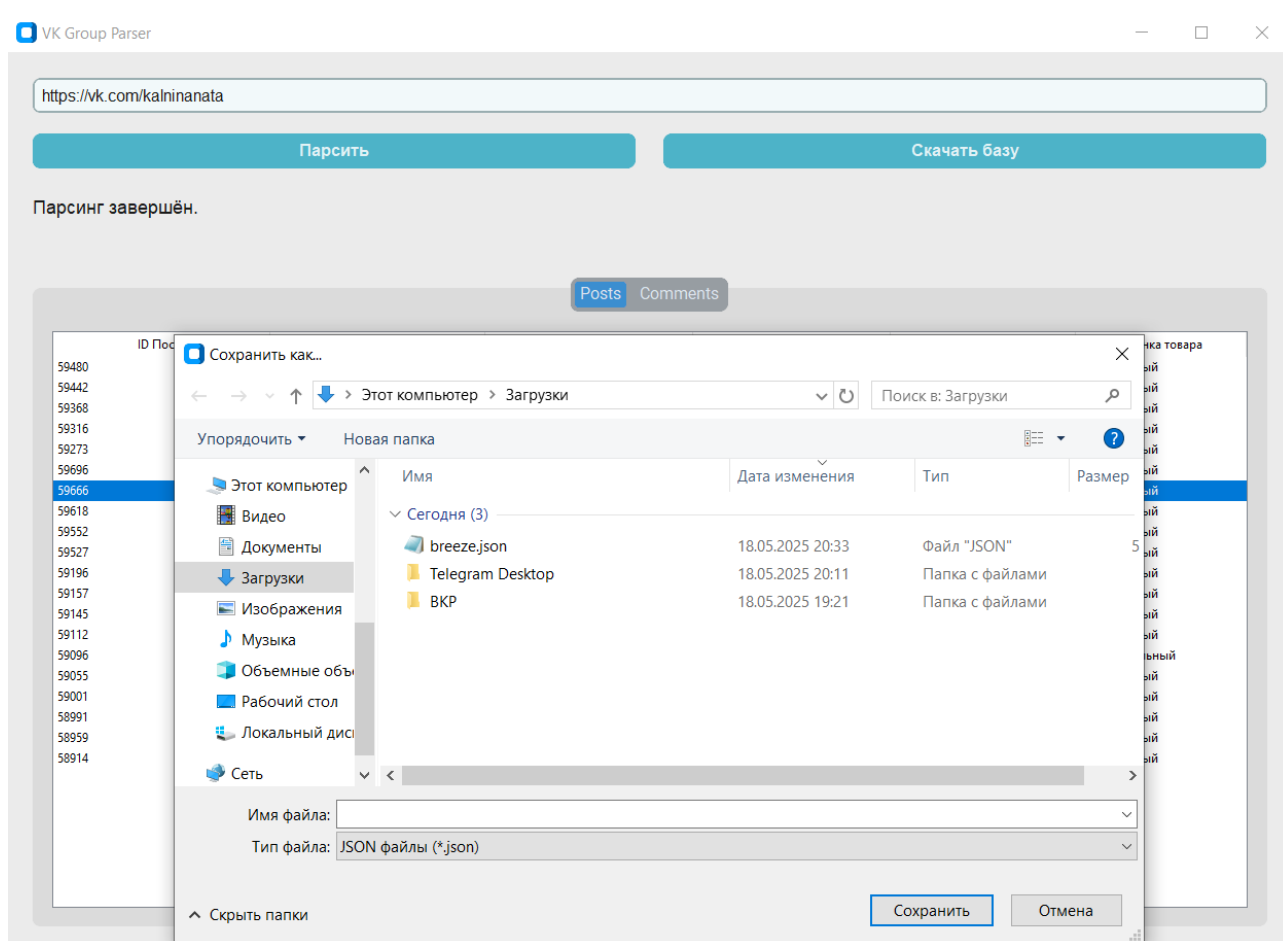


Рисунок 4.6 — Работа механизма сохранения данных

4.3 Реализация нейросетевого обучения модели

Подготовка данных начинается с загрузки таблицы в формате Excel, содержащей текстовые отзывы и соответствующие им метки: "позитивный",

"нейтральный" или "отрицательный". Прежде чем приступить к обучению, данные проходят серию предобработок. Во-первых, из таблицы удаляются мусорные или ошибочные строки, например, содержащие значение #VALUE!. Затем текстовые метки тональности конвертируются в числовой формат, чтобы обеспечить совместимость с функцией потерь: 0 — отрицательный, 1 — нейтральный, 2 — позитивный. Это представление облегчает обработку модели и позволяет корректно рассчитывать ошибку на каждом этапе обучения.

Особое внимание уделено проблеме дисбаланса классов. Социальные данные, как правило, содержат неравномерное распределение по тональностям — например, позитивных отзывов может быть существенно больше, чем негативных. Чтобы минимизировать влияние этого перекоса, были рассчитаны веса классов на основе их частотности в выборке. Эти веса затем передаются в функцию потерь CrossEntropyLoss, что позволяет модели не смещаться в сторону наиболее часто встречающихся классов, сохраняя чувствительность к менее представленным меткам, особенно к негативным отзывам.

В архитектурном плане модель реализуется как отдельный модуль BertClassifier, обертывающий предобученный BERT и дополняющий его линейным классификатором на 3 выхода, соответствующих трём категориям. Перед подачей данных в модель они токенизируются с помощью встроенного токенизатора BertTokenizer, приводятся к фиксированной длине (максимум 32 токена) и помещаются в DataLoader с заданным размером батча (8 объектов). Это позволяет эффективно использовать память и ускоряет процесс обучения. Обучение проводится с использованием оптимизатора Adam и малым значением learning rate ($1e-5$), чтобы сохранить устойчивость и избежать перезаписи предобученных параметров модели.

В ходе тренировки используется цикл обучения, рассчитанный на 30 эпох. На каждой эпохе модель обучается на всём доступном датасете, после чего логируется общая потеря и вычисляется время прохождения эпохи. Для предотвращения переобучения и резких скачков градиентов применяется техника gradient clipping, ограничивающая норму градиента на каждом шаге.

					ПГУ 1.090501.15.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		34

Использование библиотеки `tqdm` позволяет в реальном времени отслеживать прогресс обучения и текущие значения функции потерь, что удобно при отладке и анализе динамики сходимости модели.

По завершении обучения веса модели сохраняются в отдельный файл (`bert_classifier.pth`). Это позволяет избежать повторного обучения в дальнейшем и использовать уже обученную модель для быстрых предсказаний в пользовательском интерфейсе. Таким образом, инференс становится возможным прямо из GUI, без необходимости загрузки модели заново или доступа к обучающим данным.

Подход, использованный при реализации обучения, обеспечивает модульность, масштабируемость и адаптируемость решения. При необходимости модель может быть дообучена на новых данных, доработана под другие задачи (например, бинарную классификацию или тематическое моделирование), а также использована для анализа данных из других источников за пределами VK.

4.4 Реализация серверной части приложения

Серверная часть приложения реализована на современном веб-фреймворке FastAPI, обеспечивающем высокую производительность, асинхронность и удобный REST-интерфейс. Основная цель серверной компоненты — предоставление API-эндпоинтов для парсинга данных из социальной сети VK, сохранения и обработки этих данных, а также анализа их эмоционального содержания с помощью нейросетевой модели.

4.4.1 Архитектура и взаимодействие компонентов

Серверная часть реализована в виде самостоятельного HTTP-сервиса на базе фреймворка FastAPI. Она организована таким образом, чтобы предоставить доступ к двум ключевым функциям через REST-интерфейс. Основной файл `server.py` отвечает за создание экземпляра приложения и регистрацию маршрутов, каждый из которых решает строго определённую задачу в рамках обработки данных из социальной сети ВКонтакте.

Первый маршрут — /parse — реализует получение и первичную обработку данных из заданной VK-группы. При получении запроса с доменным именем сообщества он инициирует обращение к внешнему API, собирает список постов и комментариев, после чего формирует структурированные таблицы. Далее проводится базовая очистка и нормализация данных: подсчёт лайков, фильтрация нестандартных значений, приведение полей к единому формату. Обработанные данные возвращаются клиенту в виде валидного JSON, пригодного для последующего анализа или хранения.

Второй маршрут — /posts_and_comments_analyze — предназначен для анализа тональности собранных комментариев. Он загружает данные из базы данных, где ранее были сохранены посты и комментарии, и применяет модель анализа тональности к каждому тексту. Результаты анализа — положительный, нейтральный или отрицательный — агрегируются по каждому посту, формируя обобщённую метку, отражающую общее восприятие публикации комментаторами. Эти оценки добавляются к таблице постов, дополняя исходные данные качественной аналитикой.

Каждый маршрут использует собственную модель запроса, определённую через Pydantic. Это позволяет строго типизировать входящие данные, обеспечивая автоматическую валидацию на уровне API.

Архитектура серверной части проекта показана на рисунке 4.7

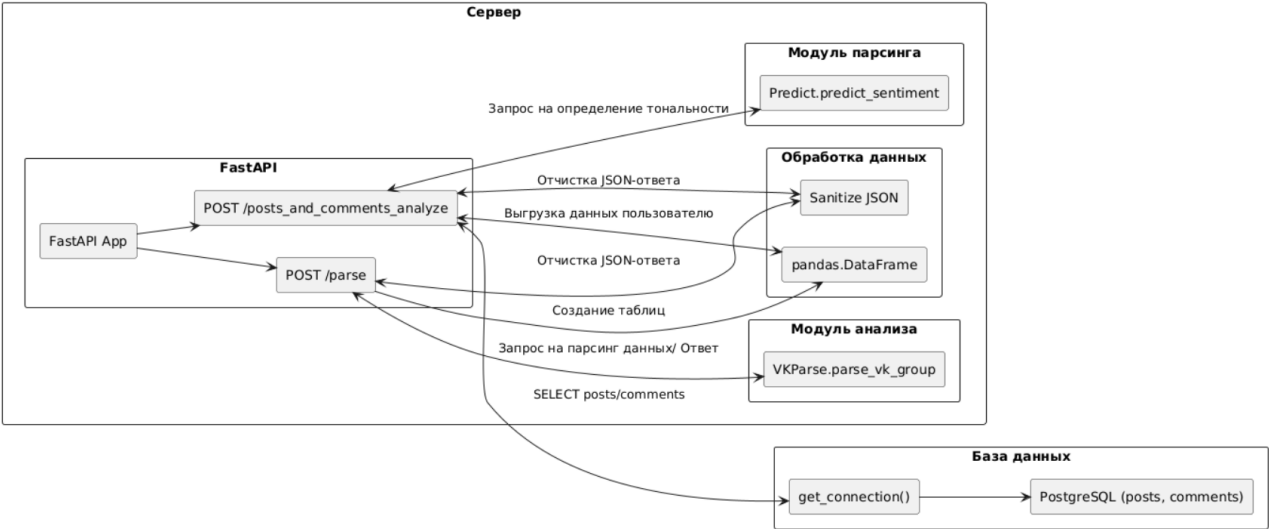


Рисунок 4.7 — Архитектура серверной части системы

4.4.2 Парсинг и запись данных

При обращении к маршруту /parse запускается чётко структурированный процесс сбора и сохранения данных из социальной сети «ВКонтакте». Он реализован через вспомогательные функции, сгруппированные в модуле VKParse, что обеспечивает чёткое разделение ответственности и облегчает модификацию или масштабирование этого этапа.

На диаграмме ниже показана последовательность взаимодействий между пользователем, сервером, VK API и базой данных в процессе парсинга и записи данных [24]:

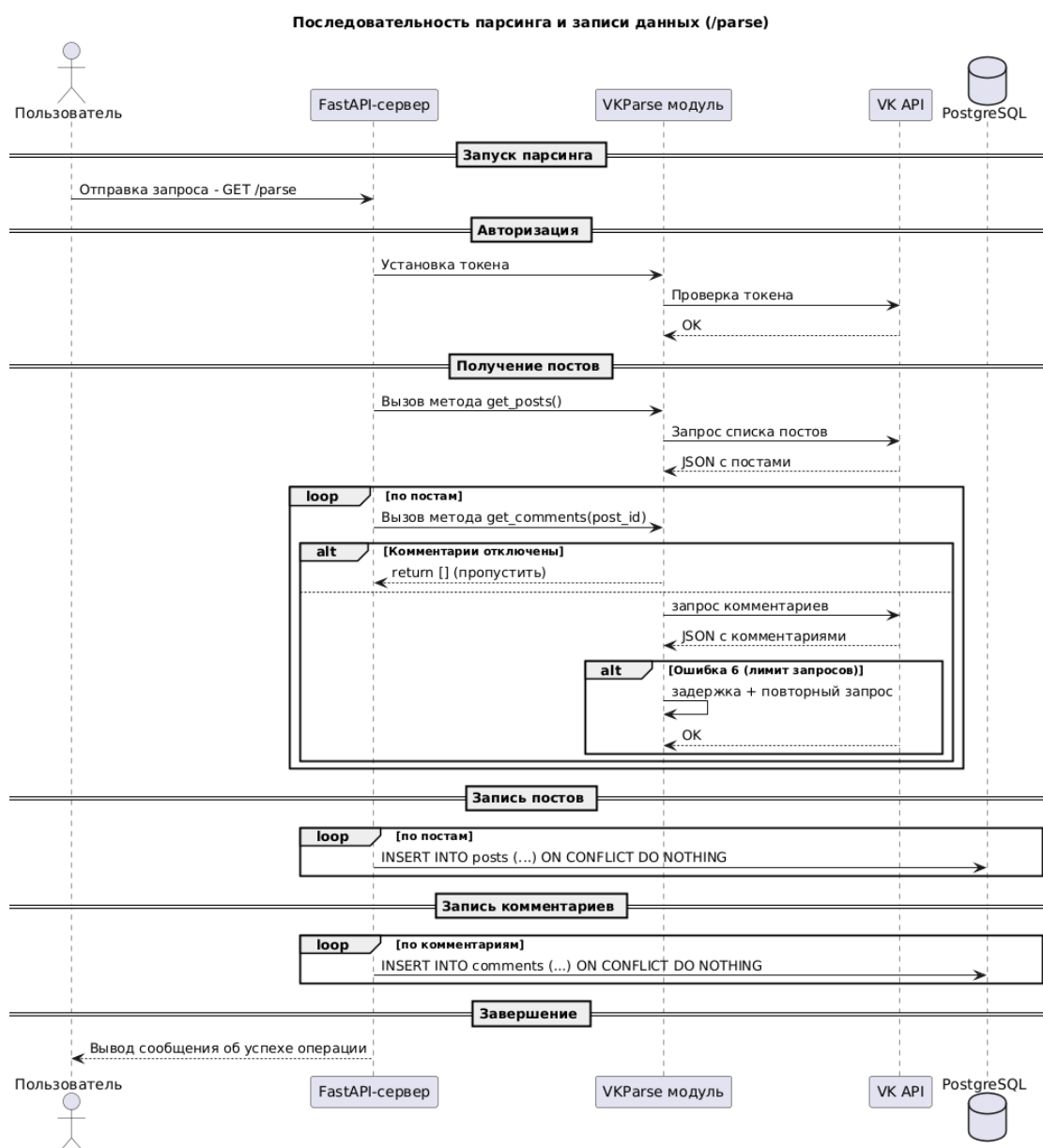


Рисунок 4.8 — Диаграмма последовательности парсинга и записи данных

Процесс начинается с авторизации через access-token пользователя VK, после чего осуществляется последовательный сбор постов с публичной стены указанного сообщества. Для каждого полученного поста дополнительно запрашиваются комментарии, при этом учитывается возможность ограничения доступа, например, полное отключение комментариев на уровне поста.

Для повышения устойчивости к ограничениям VK API реализован антиспам-механизм: каждый запрос сопровождается рандомизированной задержкой (на основе BASE_DELAY и uniform), а также автоматической повторной попыткой в случае возникновения ошибки с кодом 6 («слишком много запросов в секунду»). Такая схема обеспечивает баланс между производительностью и соблюдением лимитов API.

После сбора данных реализуется их безопасная запись в PostgreSQL. Сначала таблицы posts и comments очищаются (через DELETE), чтобы исключить дублирование и устаревшие записи. Вставка данных сопровождается конструкцией ON CONFLICT DO NOTHING, позволяющей избежать повторных записей по первичному ключу. Все данные предварительно сериализуются и проходят обработку исключений: любые сбои при подключении, записи или сериализации логируются, не нарушая работу сервера. Это критично для обеспечения надёжности в продакшене.

4.4.3 Обработка и анализ комментариев

После сбора данных маршрут /posts_and_comments_analyze инициирует подключение к PostgreSQL и извлекает ранее сохранённые записи. Если одна из таблиц пуста, сервер корректно возвращает ошибку HTTP 404, информируя пользователя об отсутствии данных.

На следующем этапе осуществляется предсказание тональности каждого комментария. Для этого вызывается функция predict_sentiment, реализованная в модуле predict, где подключена нейросетевая модель, основанная на BERT и дообученная на датасете русскоязычных отзывов. Все тексты конвертируются в числовое представление и подаются в модель, возвращающую один из трёх классов: положительный, нейтральный или отрицательный.

Затем производится агрегация оценок на уровне поста: каждый пост получает интегральную оценку, рассчитанную как округлённое среднее значение тональности всех его комментариев. Это значение сохраняется в дополнительной колонке «Оценка товара». Если у поста нет комментариев, ему присваивается статус «Нет комментариев».

4.4.4 Защита и устойчивость

На протяжении всего процесса взаимодействия с пользователем предусмотрена многоуровневая система обработки исключений: ошибки API, сетевые сбои, проблемы с базой данных, а также невозможность распознать тональность комментария корректно логируются и возвращаются клиенту в виде понятных HTTP-ответов. Внутри серверной логики применена функция универсальной очистки, которая обрабатывает возможные NaN, inf и другие недопустимые для JSON-ответа значения, обеспечивая стабильность и корректность ответов.

Диаграмма структуры обработки ошибок и защиты данных представлена на рисунке 4.9

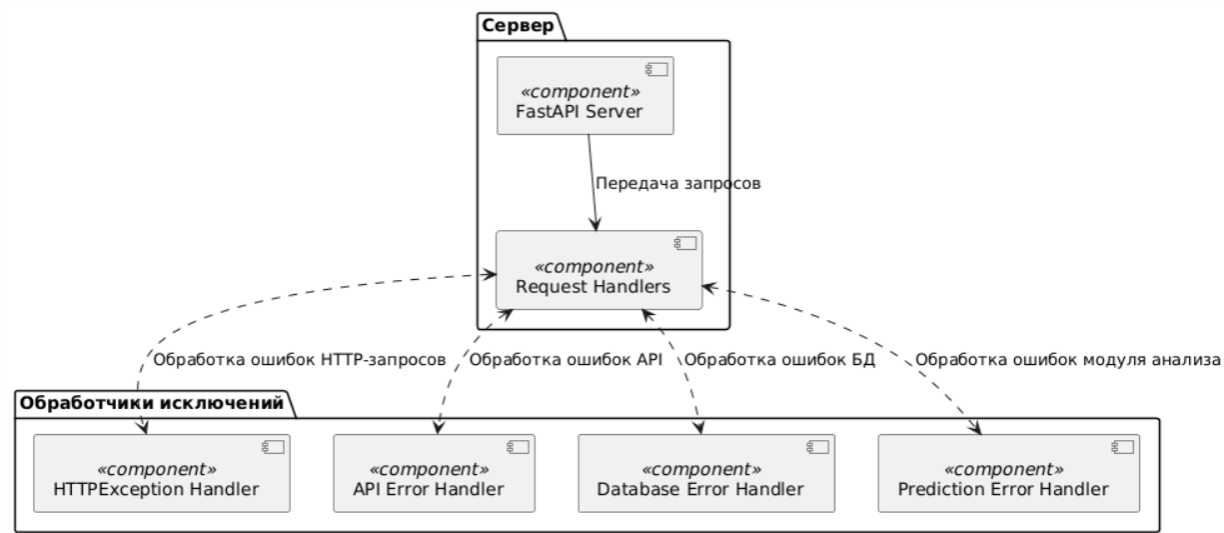


Рисунок 4.9 — Диаграмма модуля обработки ошибок

4.4.5 Интеграция и масштабируемость

Сервер построен на базе FastAPI — современного, высокопроизводительного веб-фреймворка, поддерживающего асинхронность «из коробки». Архитектура

приложения спроектирована с возможностью масштабирования и последующей интеграции в более сложную экосистему.

Масштабируемость обеспечивается через:

- Возможность развертывания как на локальных машинах (например, для разработки или анализа локальных данных), так и в облачных средах (Yandex.Cloud, Heroku, Docker, AWS и т. д.);
- Модульность архитектуры — каждый компонент (VKParse, predict, server) оформлен как самостоятельный модуль с чёткой зоной ответственности;
- Расширяемость API — маршруты легко масштабируются, можно добавлять, например, /summary, /stats, /top_comments без вмешательства в существующую логику;
- Потенциал для интеграции с очередями задач (например, Celery + Redis) и кешированием промежуточных результатов;
- Возможность перевода модели на REST- или gRPC-сервис для отделения ML-инференса от FastAPI-интерфейса.

Таким образом, серверная часть выполняет роль гибкой, надёжной и промышленно готовой основы для сбора, хранения и анализа пользовательского контента из VK. Благодаря использованию современных технологий — от BERT до FastAPI и PostgreSQL — она может быть адаптирована для других источников, других моделей анализа и любых клиентских интерфейсов.

5 Проектирование базы данных

5.1 Разработка структур данных

На основе анализа предметной области были разработаны следующие структуры данных:

- POSTS;
- COMMENTS;

Структуры данных для накопителей «POSTS», «COMMENTS» приведены в таблицах 5.1 – 5.2.

Таблица 5.1 - Описание структуры данных POSTS

Имя атрибута	Название
post_id	Идентификатор поста
owner_id	Идентификатор владельца стены
likes	Количество лайков
text	Текст поста

Таблица 5.2 - Описание структуры данных COMMENTS

Имя атрибута	Название
comment_id	Идентификатор комментариев
post_id	Идентификатор постов
author	Псевдоним автора комментария
text	Содержимое комментария
likes	Количество лайков

Таким образом, разработаны структуры данных для автоматизированной системы определения рейтинга товаров на основе нейросетевого анализа комментариев в vk.

5.2 Концептуальное проектирование

Концептуальная модель данных отображает обобщающее представление о данных, независимое от типа выбранной СУБД. Она описывает то, какие данные хранятся в базе данных, а также связи, существующие между ними.

На основе анализа предметной области с помощью CASE – средства OpenModelSphere 3.2 разработана концептуальная модель данных.

Концептуальная модель данных [25] представлена на рисунке 4.3 и содержит 8 сущностей.

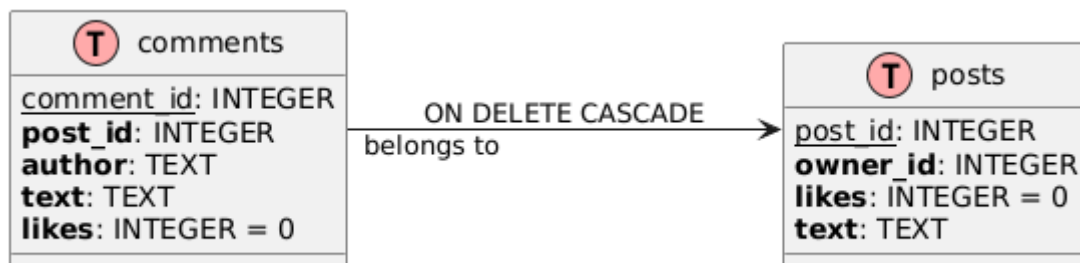


Рисунок 5.1 - Концептуальная модель данных

Описание сущностей и их атрибутов приведено в таблицах 5.3 – 5.4.

Таблица 5.3 - Описание атрибутов сущности POSTS

Имя атрибута	Название	Первичный ключ
post_id	Идентификатор поста	РК
owner_id	Идентификатор владельца стены	
likes	Количество лайков	
text	Текст поста	

Таблица 5.4 - Описание атрибутов сущности COMMENTS

Имя атрибута	Название	Первичный ключ
comment_id	Идентификатор комментариев	РК
post_id	Идентификатор постов	
author	Псевдоним автора комментария	
text	Содержимое комментария	
likes	Количество лайков	

Сущность POSTS связана с сущностью COMMENTS связью с показателем кардинальности 0,N - 1,1, так как пост может содержать либо несколько

комментариев, либо ни одного, в то время как один комментарий привязан к одному конкретному посту. Степень участия связи между сущностями полная.

Таким образом, разработана концептуальная модель данных.

5.3 Логическое проектирование

На основе концептуальной модели данных была разработана логическая (реляционная) модель данных.

Реляционная модель данных представлена на рисунке 5.2 и содержит 8 сущностей.

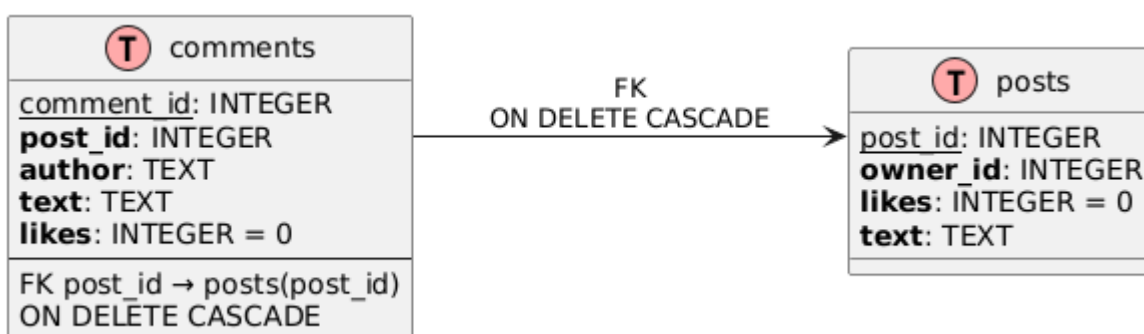


Рисунок 5.2 - Логическая модель данных

Описание сущностей и их атрибутов приведено в таблицах 5.6 – 5.7.

Таблица 5.6 - Описание атрибутов сущности POSTS

Имя атрибута	Название	Первичный ключ	Внешний ключ
post_id	Идентификатор поста	РК	
owner_id	Идентификатор владельца стены		
likes	Количество лайков		
text	Текст поста		

Таблица 5.7 - Описание атрибутов сущности COMMENTS

Имя атрибута	Название	Первичный ключ	Внешний ключ
comment_id	Идентификатор комментариев	РК	
post_id	Идентификатор постов		FK
author	Псевдоним автора комментария		
text	Содержимое комментария		
likes	Количество лайков		

База данных находится в 1НФ, если на пересечении каждой строки и столбца содержится только одно значение (отсутствуют повторяющиеся группы данных).

База данных, логическая модель которой представлена на рисунке 5.2, находится в 1НФ. Для подтверждения того, что она находится в 1НФ необходимо рассмотреть каждую сущность:

- сущность POSTS имеет ключевой атрибут post_id. Атрибуты owner_id, likes, text данной сущности являются простым и неделимым;

- сущность COMMENTS имеет ключевой атрибут comment_id. Атрибуты post_id, author, text, likes данной сущности являются простыми и неделимыми.

База данных находится во 2НФ, если она находится в 1НФ и каждый атрибут, не входящий в состав первичного ключа, характеризуется полной функциональной зависимостью от этого первичного ключа.

База данных, логическая модель которой представлена на рисунке 5.2, находится во 2НФ, так как она находится в 1НФ и все неключевые атрибуты всех сущностей функционально полно зависят от первичных ключей этих сущностей.

Сущность POSTS находится во 2НФ, так как имеет единственный ключевой атрибут post_id. Не ключевые атрибуты owner_id, likes, text данной сущности функционально полно зависят от данного первичного ключа. Следовательно, можно сделать вывод о том, что данная сущность находится во 2НФ.

Сущность COMMENTS находится во 2НФ, так как имеет единственный ключевой атрибут comment_id. Не ключевые атрибуты post_id, author, text, likes данной сущности функционально полно зависят от данного первичного ключа. Следовательно, можно сделать вывод о том, что данная сущность находится во 2НФ.

База данных находится в 3НФ, если она находится в 1 и 2 нормальных формах и не имеет не входящих в первичный ключ атрибутов, которые находились бы в транзитивной зависимости от этого первичного ключа.

Ни в одной из сущностей логической модели данных, представленной на рисунке 5.2, не существует транзитивных зависимостей.

В сущности POSTS не ключевые атрибуты owner_id, likes, text и post_id не зависят друг от друга. Следовательно, можно сделать вывод о том, что данная сущность логической модели находится в 3НФ.

В сущности COMMENTS не ключевые атрибуты post_id, author, text, likes и comment_id не зависят друг от друга. Следовательно, можно сделать вывод о том, что данная сущность логической модели находится в 3НФ.

Таким образом, разработана логическая модель данных.

5.4 Физическое проектирование

Физическая модель данных — это логическая модель данных, учитывающая особенности специфической целевой СУБД.

Созданная на этапе логического проектирования диаграмма «сущность-связь», на этапе физического проектирования была преобразована с помощью системы Open Modelsphere 3.2 в реляционную модель. Физическая модель данных представлена на рисунке 5.3.

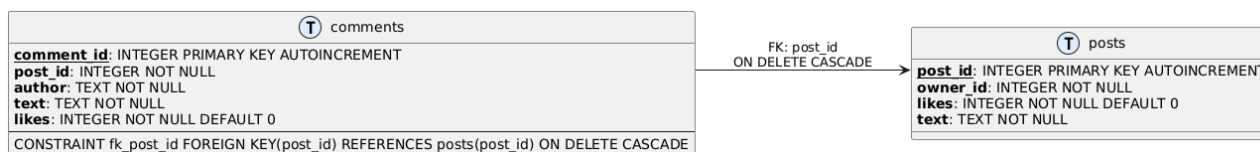


Рисунок 5.3 - Физическая модель данных

На этапе физического проектирования были определены типы данных атрибутов сущностей. Описание сущностей и их атрибутов приведено в таблицах 5.8 – 5.9.

Таблица 5.8 - Описание атрибутов сущности POSTS

Имя атрибута	Физическое имя	Ключевое значение	Нулевое значение	Тип данных
post_id	post_id	PK	Недопустимо	INTEGER
owner_id	owner_id		Недопустимо	INTEGER
likes	likes		Недопустимо	INTEGER
text	text		Недопустимо	TEXT

Таблица 5.9 - Описание атрибутов сущности COMMENTS

Имя атрибута	Физическое имя	Ключевое значение	Нулевое значение	Тип данных
comment_id	comment_id	PK	Недопустимо	INTEGER
post_id	post_id		Недопустимо	INTEGER
author	author		Недопустимо	TEXT
text	text		Недопустимо	TEXT
likes	likes		Недопустимо	INTEGER

Таким образом, на этапе физического проектирования разработана физическая модель данных для СУБД PostgreSQL.

6 Программа и методика испытаний

6.1 Объект испытаний

Объектом испытаний является автоматизированная информационная система определения рейтинга товаров на основе нейросетевого анализа комментариев в vk, состоящая из базы данных и приложения работы с ней. Исполняемый файл – VK_Analyzer.exe. База данных PostgreSQL хранится в виде набора служебных файлов в системной директории сервера PostgreSQL.

6.2 Цель испытаний

Испытания проводятся с целью проверки работоспособности и надежности программы VK_Analyzer.exe. Для проверки правильности работы необходимо испытать его на тестовых примерах. Сравнив полученные результаты, можно определить правильность работы программы.

6.3 Требования к программе

Разрабатываемое приложение должно обеспечивать выполнение следующих функций:

- парсинг данных выбранного сообщества «Вконтакте»;
- нейросетевой анализ тональности;
- раскрытие поста для детального рассмотрения;
- Скачивание файла в формате JSON и EXCEL;

6.4 Требования к программной документации

Программное обеспечение должно сопровождаться пояснительной запиской, которая должна содержать:

- техническое задание;
- описание программы;
- программу и методику испытаний;
- текст программы.

Структура и содержание программной документации должны соответствовать требованиям стандартов единой системы программной документации (ЕСПД)

6.5 Средства и порядок испытаний

Тестирование разработанного приложения проводилось на основе тестовых примеров, разработанных для проверки каждой функции системы.

Во время испытаний использовался персональный компьютер со следующими характеристиками:

- ОС Windows 10;
- тактовая частота процессора – 3400 МГц;
- оперативная память – 16 ГБ;
- объем жесткого диска – 512 ГБ.

Программные средства, используемые во время испытаний:

- СУБД PostgreSQL 17.

Был выбран следующий порядок испытаний:

- запустить программу на выполнение;
- провести тестирование программы по тестам;
- сравнить реакции программы с ожидаемым результатом;
- сделать выводы по результатам тестирования о работоспособности

программы.

Приложение работает верно, если результаты его работы совпадают с результатами выполненных тестовых примеров.

6.6 Методика испытаний

В процессе проведения испытаний были получены результаты, описанные ниже.

Тестовый пример 1. Проверка выполнения функции парсинга данных выбранного сообщества и нейросетевого анализа тональности.

Для проверки правильности работы программы необходимо было запустить программу на выполнение, используя откомпилированный файл VK_Analyzer.exe. В результате запуска было открыто основная страница приложения (рисунок Б.1). На главной форме пользователю необходимо вставить в текстовую строку ссылку на группу с указанием домена.

После этого пользователю будет выведена вся информация выбранного им домена (рисунок Б.2).

Тестовый пример 2. Проверка выполнения функции раскрытия поста для детального рассмотрения.

Для проверки правильности работы программы необходимо было запустить программу на выполнение, используя откомпилированный файл VK_Analyzer.exe. В результате запуска было открыто основная страница приложения (рисунок Б.1). На главной форме пользователю необходимо вставить в текстовую строку ссылку на группу с указанием домена.

После этого пользователю будет выведена вся информация выбранного им домена (рисунок Б.2). Нажав дважды по интересующей записи поста сообщества, откроется форма с детальным описанием выбранной записи (рисунок Б.3) и возможностью перехода на страницу поста по ссылке (рис. Б.4).

Тестовый пример 3. Проверка выполнения функции скачивание файла в формате JSON и EXCEL.

Для проверки правильности работы программы необходимо было запустить программу на выполнение, используя откомпилированный файл VK_Analyzer.exe. В результате запуска было открыто основная страница приложения (рисунок Б.1). На главной форме пользователю необходимо вставить в текстовую строку ссылку на группу с указанием домена.

После этого пользователю будет выведена вся информация выбранного им домена (рисунок Б.2). Нажав на кнопку «Скачать базу» будет выведено диалоговое окно с выбором имени файла и его форматом: JSON и EXCEL (рисунок Б.5). Далее пользователю остается только нажать кнопку «Сохранить» и файл сохранится в выбранной папке.

7 Оценка затрат на разработку автоматизированной системы определения рейтинга товаров на основе нейросетевого анализа комментариев

7.1 Организация производства

Разработка автоматизированной системы анализа пользовательских комментариев во «ВКонтакте» с целью выявления тональности и формирования аналитики по восприятию товаров предполагает реализацию программного решения, ориентированного на сбор, обработку, классификацию и визуализацию мнений пользователей. Основная цель проекта — предоставление эффективного инструмента для анализа репутации товаров и поддержки управленческих решений в области маркетинга.

Создание программной системы осуществляется проектной командой. Для обеспечения продуктивной работы арендуется офисное помещение площадью 40 м² по цене 35 000 рублей в месяц. В арендную плату включена базовая офисная мебель — столы, кресла, стеллажи и шкафы для хранения техники и документов.

Для реализации проекта необходимо закупить современные ноутбуки, соответствующие требованиям по производительности для работы с алгоритмами машинного обучения и анализа текста, а также обеспечить подключение к стабильной сети интернет и облачным сервисам хранения данных. Также планируется установка лицензионного программного обеспечения (инструменты для программирования, библиотеки анализа текста, СУБД) и приобретение расходных материалов. Сведения о требуемой технике и ресурсах представлены в таблице 7.1.

Таблица 7.1 – Остаточная стоимость оборудования

Наименование оборудования	Источник	Количество	Остаточная стоимость за единицу, рублей	Потребляемая мощность, Вт
Ноутбук Lenovo ThinkBook 15 G4 ABA	Интернет-магазин цифровой и бытовой техники «DNS»	6	65 000	45

Наименование оборудования	Источник	Количество	Остаточная стоимость за единицу, рублей	Потребляемая мощность, Вт
Маршрутизатор TP-Link Archer AX10	Интернет-магазин цифровой и бытовой техники «Ситилинк»	1	4200	12

Для разработки системы использовалось лицензионное программное обеспечение, представленное в таблице 7.2.

Таблица 7.2 — Потребность в лицензионном программном обеспечении

Лицензионное программное обеспечение	Источник	Цена, руб.	Срок действия лицензии, дней
Microsoft Windows 10 Professional	Интернет-магазин цифровой и бытовой техники «DNS»	Входит в стоимость ПК	Неограниченно
Microsoft Visual Studio Code 1.99.3	Официальный сайт среды разработки Microsoft Visual Studio Code https://code.visualstudio.com/	Бесплатно для индивидуального и малого коммерческого использования	Неограниченно
PostgreSQL 17	Официальный сайт СУБД PostgreSQL https://www.postgresql.org	Бесплатно, лицензия PostgreSQL License	Неограниченно
OpenOffice Writer	Официальный сайт офисного пакета Apache OpenOffice https://www.openoffice.org	Бесплатно, лицензия Apache License 2.0	Неограниченно
Open ModelSphere 3.2.2	Официальный сайт инструмента для моделирования данных и UML — ModelSphere https://www.modelsphere.org	Бесплатно, лицензия GPL	Неограниченно

Для обеспечения нормальной работы разработчиков будут приобретены дополнительные (расходные) материалы, представленные в таблице 7.3.

Таблица 7.3 – Потребность в дополнительных материалах

Материал	Источник	Количество	Ориентировочная цена за ед., руб.
Ручки гелевые Pilot G-2	Интернет-магазин канцтоваров «ОфисМаркет»	10	60
USB-флешка Kingston 32 ГБ	Интернет-магазин цифровой техники «DNS»	6	900

Разработка программного продукта осуществляется командой проекта, в состав которой входят:

- руководитель проекта;

- программисты;
- Data Scientist;
- ML-инженер;
- тестировщик.

Описание состава коллектива, количественного соотношения специалистов и их функциональных обязанностей приведено в таблице 7.4.

Таблица 7.4 — Состав коллектива разработчиков

Специалист	Количество	Описание занятия
Ведущий разработчик	1	Руководство процессом разработки, архитектура системы, контроль качества
Программист	2	Разработка кода, реализация функционала анализа и обработки данных
Аналитик данных	1	Сбор, обработка и анализ данных товаров, подготовка отчетов
Тестировщик	1	Тестирование программного обеспечения, выявление и документирование ошибок
Системный администратор	1	Настройка серверов, обеспечение безопасности и стабильности работы системы

Данные по основной заработной плате персонала и их количественному составу представлены в таблице 7.5.

Таблица 7.5 — Занятость и заработная плата персонала

Специалист	Количество	Сумма з/п, руб.
Ведущий разработчик	1	180 000
Программист	2	120 000
Аналитик данных	1	130 000
Тестировщик	1	100 000
Системный администратор	1	110 000

Заработная плата каждого участника проекта имеет следующее обоснование:

Ведущий разработчик получает заработную плату в размере 180 000 рублей. Это обосновано его ключевой ролью в проекте: он отвечает за проектирование архитектуры системы, принятие технических решений, а также за координацию работы всей команды. Учитывая высокую квалификацию и опыт, а также рыночные ставки по стране, данное значение является обоснованным.

Программисты, каждый с заработной платой 120 000 рублей, занимаются реализацией программного обеспечения, включая написание кода, интеграцию с

базами данных и реализацию логики анализа товарных данных. Их уровень оплаты соответствует специалистам со средней квалификацией и опытом от 2 лет, что позволяет эффективно и в срок выполнить технические задачи проекта.

Аналитик данных получает 130 000 рублей. Он занимается обработкой товарной информации, построением аналитических моделей, формированием отчетов и выводов, необходимых для обоснованного принятия решений. Эта сумма соответствует среднерыночной оплате труда специалиста в области анализа данных с хорошей подготовкой и практическим опытом.

Тестировщик, получающий 100 000 рублей, отвечает за проверку корректности работы программного продукта, выявление и фиксацию ошибок, а также обеспечение надежности и стабильности анализа данных. Уровень оплаты адекватен средней рыночной стоимости труда специалиста по качеству в IT-проектах.

Системный администратор с оплатой 110 000 рублей выполняет задачи по поддержке IT-инфраструктуры проекта: настройке оборудования, программного обеспечения, обеспечению безопасности и бесперебойной работы рабочих станций. Его зарплата соответствует средней по рынку для данной должности с необходимыми компетенциями.

7.2 Календарное планирование разработки

Календарное планирование является одним из ключевых этапов в управлении жизненным циклом проекта по созданию автоматизированной информационной системы. Его основная задача — структурировать процесс разработки во времени, чтобы обеспечить последовательность выполнения задач, равномерную загрузку команды и достижение результата в обозначенные сроки.

В рамках планирования определяются все составляющие проекта: от перечня стадий и этапов работ до необходимого количества специалистов, их загрузки и продолжительности участия на каждом этапе. Такой подход позволяет минимизировать риски задержек, упростить контроль прогресса и создать прозрачную дорожную карту реализации проекта.

Календарный план должен четко отражать:

- последовательность стадий и этапов проекта;
- сроки начала и завершения каждого этапа;
- длительность каждой задачи;
- занятость специалистов по ролям и объему на всем протяжении проекта.

Ниже приведена таблица, демонстрирующая распределение задач, ресурсов и времени реализации проекта. Общая продолжительность разработки системы рассчитана в рабочих днях и представлена в итоговой строке.

Таблица 7.6 – Стадии и этапы реализации проекта

Название стадии	Название этапа	Разработчик	Количество разработчиков	Длительность работы, раб.дни	Длитель ность этапа, раб.дни	Длитель ность стадии, раб.дни
1 Исследование и подготовка	1.1 Анализ предметной области	Ведущий разработчик	1	8	8	18
		Аналитик данных	1	8		
	1.2 Формулировка требований к системе	Ведущий разработчик	1	6	6	
		Аналитик данных	1	6		
	1.3 Формирование требований	Ведущий разработчик	1	4	4	
		Аналитик данных	1	4		
2 Разработка архитектуры	2.1 Проектирование логической архитектуры	Ведущий разработчик	1	5	5	10
		Аналитик данных	1	5		
	2.2 Подготовка инфраструктуры	Ведущий разработчик	1	5	5	
		Системный администратор	1	5		
3 Реализация	3.1 Программирование интерфейса и логики	Ведущий разработчик	1	16	16	20
		Программист	2	16		
	3.2 Настройка серверной среды	Ведущий разработчик	1	4	4	
		Системный администратор	1	4		
4 Внутреннее тестирование	4.1 Проверка функционала	Ведущий разработчик	1	5	5	9
		Тестировщик	1	5		
	4.2 Отладка и исправление багов	Ведущий разработчик	1	4	4	
		Программист	2	4		
		Тестировщик	1	4		
5 Финальное тестирование и сдача	5.1 Интеграционное тестирование	Ведущий разработчик	1	3	3	5
		Тестировщик	1	3		
	5.2 Приемка и сдача проекта	Ведущий разработчик	1	2	2	

Название стадии	Название этапа	Разработчик	Количество разработчиков	Длительность работы, раб.дни	Длитель ность этапа, раб.дни	Длитель ность стадии, раб.дни
		Аналитик данных	1	2		
		Тестирующий	1	2		
Итого						62

Календарный план реализации проекта представлен в таблице 7, где значение «Итого» позволяет сделать вывод о планируемой длительности реализации проекта, выраженной в количестве календарных дней.

Таблица 7.7 – Календарный план

Название стадии	Название этапа	Объёмы работ на стадии, %	Дата работ		Количес тво дней	Отчетные документы
			Начало	Окончание		
1 Исследование и подготовка	1.1 Анализ предметной области	11	24.02.25	5.03.25	10	Описание предметной области
	1.2 Формулировка требований к системе	9	6.03.25	13.03.25	8	Технико- экономическое обоснование разработки
	1.3 Формирование требований	7	14.03.25	19.03.25	6	Документ требований
2 Разработка архитектуры	2.1 Проектирование логической архитектуры	8	20.03.25	26.03.25	7	Техническое задание
	2.2 Подготовка инфраструктуры	8	27.03.25	02.04.25	7	
3 Реализация	3.1 Программирование интерфейса и логики	25	03.04.25	24.04.25	22	Программный код
	3.2 Настройка серверной среды	7	25.04.25	30.04.25	6	Акт о настройке и вводе в эксплуатацию серверной среды
4 Внутреннее тестирование	4.1 Проверка функционала	10	5.05.25	13.05.25	9	Отчет о результатах функционального тестирования
	4.2 Отладка и исправление багов	7	14.05.25	19.05.25	6	Журнал регистрации и устранения ошибок (баг-репорт)
5 Финальное тестирование и сдача	5.1 Интеграционное тестирование	3	20.05.25	22.05.25	3	Протокол интеграционного тестирования
	5.2 Приемка и сдача проекта	5	23.05.25	26.05.25	4	Акт приемки-сдачи выполненных работ
Итого		100	24.02.25	26.05.25	88	Утвержденная проектная и программная документация

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090501.15.001 ПЗ

Лист

55

Разработка автоматизированной информационной системы запланирована на период с 3 марта 2025 года по 26 мая 2025 года. Общая продолжительность календарного плана с учётом выходных и праздничных дней составляет 88 календарных дней, из которых непосредственно на выполнение проектных и программных работ отведен 62 рабочий день.

Разработанный календарный план представлен в виде диаграммы Ганта на рисунке 7.1.

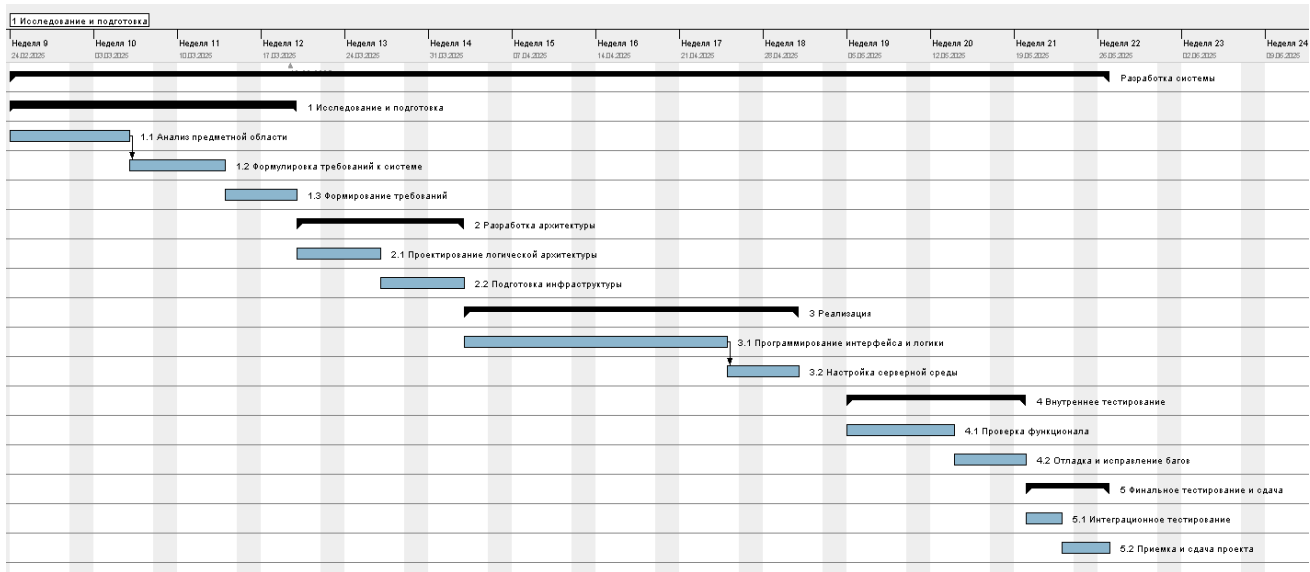


Рисунок 7.1 — Диаграмма Ганта

В соответствии с диаграммой Ганта длина критического пути разработки программы составила 62 день, что соответствует минимальному количеству времени, необходимому для разработки автоматизированной информационной системы определения рейтинга товаров на основе нейросетевого анализа комментариев в vk. С учетом выходных дней при 5-тидневной рабочей неделе по 8 часов в сутки длительность разработки составит 88 календарных дней. Начало разработки — 24.02.2025, окончание — 26.05.2025.

7.3 Расчет себестоимости программного обеспечения

Себестоимость разработки программного продукта включает следующие статьи затрат:

- затраты на приобретение оборудования;
- затраты на приобретение лицензионного программного обеспечения;
- затраты на приобретение дополнительных материалов;
- затраты на заработную плату;
- затраты на обслуживание.

Затраты на оборудование в период разработки составят 394 200 рублей и представлены в таблице 7.1.

Затраты на программное обеспечение не предусмотрены (см. таблицу 7.2).

Затраты на дополнительные материалы в период разработки составят 6 000 рублей и представлены в таблице 7.3.

Затраты на содержание помещений составят 105 000 рублей.

Затраты на заработную плату включают затраты на основную и дополнительную заработную плату разработчиков программного обеспечения, а также отчисления на социальные взносы.

Затраты на основную заработную плату разработчиков определяются исходя из установленной заработной платы, количества специалистов и их занятости в процессе разработки программного продукта (продолжительности работы в днях).

Дополнительная заработная плата разработчиков составляет 15% от основной заработной платы разработчиков.

С учетом выходных дней при 5-дневной рабочей неделе количество рабочих дней в месяце в среднем составляет 21 день.

Затраты на основную и дополнительную заработную плату разработчиков представлены в таблице 7.8.

Таким образом, затраты на основную заработную плату разработчиков составят 827 999,66 рублей.

Отчисления на страховые взносы – обязательные отчисления по нормам, установленным законодательством государственного социального страхования.

Таблица 7.8 – Затраты на основную и дополнительную заработную плату разработчиков

Исполнитель	Количество	Период занятости, дн.	Основная з/плата, руб.	Дополнительная з/плата, руб.	Всего
Ведущий разработчик	1	62	507 272,73	76 090,91	583 363,64
Программист	2	20	218 181,82	32 727,27	250 909,09
Аналитик данных	1	25	147 727,27	22 159,09	169 886,36
Тестировщик	1	14	63 636,36	9 545,45	73 181,81
Системный администратор	1	9	45 000,00	6 750,00	51 750,00
Итого	6	62	981 818,18	147 272,72	1 129 090,90

Страховые взносы взимаются согласно документу «Налоговый кодекс Российской Федерации (часть вторая)» от 05.08.2000 N 117-ФЗ (ред. от 28.12.2024) [26].

Действующие тарифы на страховые взносы установлены статьей 34 НК РФ и составляют:

- ОПС (обязательное пенсионное страхование) - 22 %;
- ВНиМ (обязательное социальное страхование на случай временной нетрудоспособности и в связи с материнством) - 2,9 %;
- ОМС (обязательное медицинское страхование) - 5,1 %.

Данные по суммам отчислений на страховые взносы представлены в таблице 7.9.

Таблица 7.9 – Отчисления на страховые взносы

Исполнитель	З/п всего, руб.	Отчисления в ОПС, руб.	Отчисления в ВНиМ, руб.	Отчисления в ОМС, руб.	Общая сумма отчислений, руб.
Ведущий разработчик	583 363,64	128 339,99	16 917,55	29 752,54	175 010,08
Программист	250 909,09	55 200,00	7 276,36	12 805,36	75 281,82
Аналитик данных	169 886,36	37 375,00	4 926,70	8 664,20	50 965,90
Тестировщик	73 181,81	16 100,00	2 122,27	3 731,27	21 953,54

Исполнитель	З/п всего, руб.	Отчисления в ОПС, руб.	Отчисления в ВНиМ, руб.	Отчисления в ОМС, руб.	Общая сумма отчислений, руб.
Системный администратор	51 750,00	11 385,00	1 501,75	2 639,25	15 526,00
Итого	1 129 090,90	248 399,99	32 744,63	57 592,62	338 737,24

Затраты на заработную плату с учетом затрат на основную и дополнительную заработную плату разработчиков программного продукта и отчислений на страховые взносы представлены в таблице 7.10.

Таблица 7.10 – Затраты на заработную плату разработчиков

Исполнитель	Количество	Затраты на з/п, руб.	Отчисления на страховые взносы, руб.	Всего, руб.
Ведущий разработчик	1	583 363,64	175 010,08	758 373,72
Программист	2	250 909,09	75 281,82	326 190,91
Аналитик данных	1	169 886,36	50 965,90	220 852,26
Тестирующий	1	73 181,81	21 953,54	95 135,35
Системный администратор	1	51 750,00	15 526,00	67 276,00
Итого	6	1 129 090,90	338 737,34	1 467 828,24

Затраты на заработную плату разработчиков программного обеспечения составляют 1 467 828,24 рублей, из которых сумма отчислений на страховые взносы составит 338 737,34 рублей.

Затраты на обслуживание включают следующие статьи:

- амортизация оборудования, составляющая 20% от стоимости оборудования;
- обслуживание оборудования, составляющее 15% от амортизационных отчислений;
- затраты на электроэнергию, включающие затраты на электроэнергию персональных компьютеров, оргтехники и бытовых приборов;
- аренда офиса и расходы на его содержание;
- накладные расходы, составляющие 80% от затрат на основную и дополнительную заработную плату разработчиков.

В проекте используются 6 ноутбуков стоимостью 65 000 рублей каждый и маршрутизатор стоимостью 4200 рублей каждый. Общая стоимость оборудования составляет 394 200 руб. Таким образом, затраты на амортизацию оборудования в месяц составят 6 570 рублей.

Обслуживание оборудования составляет 15% от амортизационных отчислений. Таким образом затраты на обслуживание оборудования составят 985,5 рублей.

Предположительно в день по 8 часов будут работать 6 ноутбуков.

Таким образом за 62 дня разработки ПО будет использовано 193 кВт электроэнергии на ноутбуки, а также 0,5 кВт на прочие приборы. Итого выходит 193,5 кВт.

Накладные расходы составляют 80% от заработной платы. Таким образом, затраты на накладные расходы составят 903 272,72 рублей.

Затраты на обслуживание приведены в таблице 7.11.

Таблица 7.11 – Затраты на обслуживание

Статья затрат	Показатель отчислений	Сумма отчислений в месяц, руб.	Итоговая сумма отчислений за период разработки, руб.
Амортизация оборудования	20 %	6570	18515,45
Обслуживание оборудования	15 %	985,5	2 777,32
Затраты на электроэнергию	5,19 руб. за 1 кВт/ч	355,4	1001,58
Накладные расходы	80 %	320 516,17	903 272,72
Аренда офиса и расходы на его содержание	-	35 000	105 000
Итого, руб.		363 427,07	1 030 567,07

Себестоимость разработки определяется как сумма затрат, включающая все статьи затрат на реализацию проекта, представленные в таблице 7.12.

Таблица 7.12 – Затраты на разработку программного обеспечения

Статья затрат	Сумма отчислений, руб.
Приобретение оборудования	394 200
Приобретение лицензионного программного обеспечения	-
Приобретение дополнительных материалов	6 000
Затраты на заработную плату	1 467 828,24
Затраты на обслуживание	1 030 567,07
Итого	2 898 595,31

Таким образом, сумма затрат на разработку проекта составит 2 898 595,31 рубля и будет соответствовать себестоимости программного обеспечения автоматизированной информационной системы обработки и хранения данных фотоплетизмографии.

7.4 Оценка эффективности программного обеспечения автоматизированной информационной системы определения рейтинга товаров на основе нейросетевого анализа комментариев в vk

Оценка экономической эффективности является неотъемлемым элементом обоснования необходимости реализации инвестиционного проекта. Она проводится на основе анализа соотношения предполагаемых результатов и издержек, охватывающих как операционные, инвестиционные, так и косвенные затраты, включая ресурсы, вовлеченные в поддержку, разработку и внедрение системы.

Разработка и внедрение автоматизированной информационной системы определения рейтинга товаров на основе нейросетевого анализа комментариев в социальной сети VK сопровождаются как прямыми экономическими эффектами, так и значимыми качественными результатами, которые нельзя отразить в денежных показателях, но которые существенно влияют на эффективность бизнеса.

К качественным (неисчисляемым) эффектам относятся:

- Повышение достоверности аналитических выводов благодаря использованию нейросетевых алгоритмов для оценки пользовательских комментариев, что исключает субъективность и снижает вероятность ошибок;
- Централизация и систематизация данных, позволяющая хранить всю информацию о товарных оценках в единой структуре без дублирования, с поддержкой логических связей между объектами;

- Сокращение временных затрат на ручной анализ и повышение общей скорости обработки данных при формировании товарных рейтингов;
- Упрощение процесса подготовки аналитических и отчетных документов для маркетинга, логистики или категорийного менеджмента;
- Усиление информационной безопасности за счёт четкого разграничения прав доступа к данным и внедрения механизмов авторизации;
- Поддержка обоснованных управленческих решений – на базе объективных пользовательских данных можно оперативно оценивать популярность и репутацию товаров;
- Повышение конкурентоспособности бизнеса, так как использование интеллектуального анализа отзывов даёт маркетинговое преимущество при работе с ассортиментом;
- Минимизация ошибок и искажений, связанных с человеческим фактором, при обработке больших объемов неструктурированных данных.

Таким образом, проект демонстрирует не только перспективность с точки зрения возврата инвестиций, но и высокую практическую значимость для повышения аналитического потенциала бизнеса, особенно в условиях высококонкурентной среды розничной торговли и электронной коммерции [27].

7.5 Анализ рисков при разработке и внедрении автоматизированной информационной системы определения рейтинга товаров на основе нейросетевого анализа комментариев в vk

При создании и внедрении автоматизированной информационной системы определения рейтинга товаров на основе нейросетевого анализа комментариев в социальной сети VK предусмотрен целый комплекс потенциальных проектных рисков. Они охватывают технические, программные, финансовые и организационные аспекты, однако большинство угроз сведено к минимуму благодаря реализованным стратегиям управления рисками.

Среди ключевых рисков — технические сбои, такие как неполадки в работе оборудования, нестабильность сетевых соединений или нехватка вычислительных ресурсов. Чтобы предотвратить подобные проблемы, при разработке использовались современные ноутбуки с достаточной производительностью и надёжной конфигурацией, обеспечивающей устойчивость среды разработки и тестирования.

Программные риски представлены возможными логическими ошибками в алгоритмах анализа текста, ошибками при интеграции компонентов, нестабильной работой отдельных модулей. Для повышения устойчивости архитектуры системы были использованы проверенные библиотеки и фреймворки, соответствующие требованиям по совместимости и отказоустойчивости. Кроме того, структура проекта спроектирована так, чтобы упростить отладку и поддержку: логика анализа, интерфейс и работа с данными строго разделены.

Финансовые риски контролируются посредством детального бюджетного планирования. Все необходимые статьи расходов (оборудование, зарплаты специалистов, аренда, расходные материалы) заранее рассчитаны и включены в смету. Применение инструментов календарного планирования (в том числе диаграмм и табличных планов) обеспечило точную координацию этапов и прозрачное распределение трудозатрат.

Организационные риски — такие как нехватка персонала, перегрузка отдельных специалистов или задержки по срокам — были сглажены за счёт чёткой структуры команды, в которую входят ведущий разработчик, программист, аналитик данных, тестировщик и системный администратор. Каждый специалист имеет конкретные задачи и сроки, а занятость по проекту согласована и контролируется.

Таким образом, благодаря продуманной архитектуре, использованию качественного оборудования, проверенному стеку технологий и эффективному управлению ресурсами, основные риски сведены к минимальному уровню. Это создаёт все условия для успешного завершения проекта и стабильной работы

однопользовательской автоматизированной системы в условиях реальной эксплуатации.

8 Общие принципы обеспечения безопасности на рабочем месте оператора информационной-автоматизированной системы салона красоты

8.1 Актуальность вопросов безопасности человека на рабочем месте

С расширением цифровизации и активным внедрением компьютерных технологий в трудовую деятельность всё более актуальным становится вопрос обеспечения безопасной и эргономичной среды для пользователей ПЭВМ. Почти все сотрудники в различных отраслях ежедневно взаимодействуют с вычислительными устройствами, что требует особого внимания к условиям, в которых они работают.

Непродуманная организация рабочего места, продолжительное пребывание за монитором, длительное удержание одной и той же позы, а также высокая степень зрительной и когнитивной нагрузки могут стать источником серьёзных проблем со здоровьем: от утомления глаз и болей в спине до хронических расстройств опорно-двигательного аппарата и нервной системы. Важно понимать, что безопасность оператора ПЭВМ — это не только технический, но и медицинско-психологический аспект.

Нормативные документы в области охраны труда, включая ГОСТ 12.0.003-74 [28], выделяют четыре основные группы потенциально опасных факторов:

- Физические факторы, среди которых — нарушение параметров микроклимата, неадекватное освещение, шумовое загрязнение, а также избыточная яркость или мерцание экрана;
- Психофизиологические, связанные с длительной концентрацией, зрительным перенапряжением, монотонностью задач и эмоциональным истощением;
- Биологические, включающие воздействие бактерий и грибов, особенно в помещениях с недостаточной вентиляцией;

– Химические, проявляющиеся через наличие вредных летучих соединений, выделяемых материалами оборудования или накопленных в замкнутых пространствах.

Создание безопасных условий труда при использовании ПЭВМ — это не формальность, а ключевая составляющая долгосрочной сохранности здоровья и профессиональной работоспособности. Комплексный подход к эргономике, соблюдение санитарных норм и внедрение современных стандартов позволит минимизировать негативное влияние цифровой среды на человека.

8.2 Техника безопасности при работе с электроприборами

Работа с электротехническими устройствами, включая персональные компьютеры, сетевое и периферийное оборудование, требует строгого соблюдения норм техники безопасности. Электроприборы являются источником потенциальной опасности, поскольку даже при низком напряжении возможны поражения током, короткие замыкания, возгорания и сбои в системе электропитания. Безответственное обращение с таким оборудованием может привести к несчастным случаям и угрозам для жизни и здоровья человека.

Основные требования по технике безопасности при работе с электроприборами включают следующие положения:

– Использование только исправного оборудования. Перед началом эксплуатации необходимо визуально осматривать приборы на наличие повреждений, перегретых участков, оплавленных вилок и проводов.

– Подключение к сети через сертифицированные розетки и стабилизаторы. Недопустимо подключать технику к перегруженным удлинителям, использовать неисправные или самодельные переходники.

– Запрет на работу с электроприборами мокрыми руками или в условиях повышенной влажности. Это исключает риск поражения электрическим током.

– Отключение оборудования от сети при его чистке, ремонте или длительном неиспользовании. Особенно важно полностью обесточивать технику при обслуживании или замене элементов.

– Регулярная проверка заземления и защитных автоматов. Все рабочие места должны соответствовать нормам электробезопасности по классу защиты и иметь исправное заземление.

– Соблюдение порядка прокладки кабелей и шнуров. Они не должны создавать препятствий для передвижения, быть подвержены раздавливанию или механическим повреждениям.

– Прохождение инструктажа. Все сотрудники, работающие с электрическим оборудованием, обязаны проходить вводный и периодический инструктаж по технике безопасности.

Следование этим простым, но обязательным правилам не только минимизирует риски получения травм, но и продлевает срок службы оборудования. Безопасная эксплуатация электроприборов — залог стабильной и защищённой ИТ-инфраструктуры.

8.3 Влияние ПЭВМ на физиологию человека

Повсеместное использование персональных электронных вычислительных машин (ПЭВМ) на рабочих местах оказывает комплексное воздействие на физиологию человека. Длительное пребывание за компьютером влияет сразу на несколько систем организма: зрительную, опорно-двигательную, нервную и сердечно-сосудистую. При несоблюдении норм гигиены труда и перерывов в работе это влияние может стать хроническим и привести к профессиональным заболеваниям.

Основные физиологические воздействия ПЭВМ:

– Зрительное переутомление. Монотонное рассматривание экрана приводит к зрительному напряжению, снижению частоты моргания и сухости глаз, синдрому "компьютерного зрения" (ощущение "песка" в глазах, головные боли, снижение остроты зрения). Причиной также может быть несоответствие

яркости и контраста изображения, мерцание экрана, неправильное освещение рабочего места.

– Мышечно-скелетные нарушения. Длительное сидение в статичной позе, особенно при неправильной эргономике рабочего места, вызывает напряжение в шейно-плечевом отделе, пояснице и запястьях. Это способствует развитию остеохондроза, туннельного синдрома и нарушений осанки.

– Нервно-психическое напряжение. Работа с ПЭВМ связана с высокой концентрацией внимания, необходимостью принятия решений в условиях ограниченного времени и постоянной готовности к обработке информации. Всё это вызывает умственное и эмоциональное перенапряжение, которое может выражаться в утомлении, раздражительности, нарушениях сна и повышенной тревожности.

– Нарушение кровообращения и обменных процессов. Малоподвижный образ труда замедляет периферическое кровообращение, повышает риск развития варикозного расширения вен и метаболических нарушений (в том числе ожирения), а также может способствовать развитию гиподинамии.

Профилактические меры:

- соблюдение режима труда и отдыха (перерывы каждые 60 минут работы за ПК);
- использование эргономичной мебели и корректной настройки монитора по высоте и расстоянию;
- гимнастика для глаз и физическая активность в течение рабочего дня;
- правильное освещение рабочего места;
- регулярные профилактические осмотры у офтальмолога и терапевта.

Таким образом, воздействие ПЭВМ на физиологию человека требует системного контроля и профилактики. При грамотной организации труда и соблюдении норм гигиены цифровой среды можно значительно снизить негативное влияние и сохранить здоровье работников.

8.4 Техника безопасности при работе с персональным компьютером

Согласно положениям Типовой инструкции по охране труда при работе на ПЭВМ (ТОИ Р-45-084-01), утверждённой приказом Минсвязи РФ №162 от 02.07.2001 [29], трудовая деятельность пользователей персональных компьютеров классифицируется на три группы в зависимости от характера выполняемых задач. Эта классификация определяет допустимую продолжительность работы без перерыва и необходимые профилактические меры для сохранения здоровья работников:

- Группа А — деятельность, связанная преимущественно с чтением информации с экрана. Рекомендуются два 15-минутных перерыва в течение смены, не включая обеденный перерыв.
- Группа Б — набор текстов. Работникам этой категории следует делать 10-минутный перерыв каждый час.
- Группа В — творческая работа, включающая анализ, проектирование, программирование и подобные задачи. Для данной группы рекомендованы перерывы продолжительностью 15 минут каждый час.

Физиологическое воздействие ПЭВМ проявляется в следующих аспектах:

- Повышенная нагрузка на органы зрения из-за длительной фокусировки на экране монитора.
- Мышечное перенапряжение шейно-воротниковой зоны, запястий и спины при длительном нахождении в статичной позе.
- Психозмоциональное напряжение в результате высокой концентрации внимания и плотного информационного потока.
- Для минимизации этих негативных последствий требуется:
- Регулярное проведение активных перерывов, включающих физические упражнения, зарядку, гимнастику для глаз и короткие прогулки.
- Использование качественной сертифицированной техники, отвечающей современным требованиям по эргономике и электромагнитной безопасности.

- Оптимизация рабочего времени с учётом особенностей выполняемой работы и уровня её интенсивности.
- Обеспечение правильной организации рабочего места, включая анатомически корректное кресло, соответствующий по высоте стол, правильное освещение и размещение монитора.

Необходимо подчеркнуть, что наличие современной и сертифицированной техники не освобождает работодателя от обязательств по контролю её состояния. Регулярные проверки оборудования, его настройка и обновление, а также проведение инструктажей по охране труда способствуют поддержанию безопасных условий работы.

Таким образом, соблюдение инструкций и норм, установленных профильными ведомствами, в сочетании с организационными мерами и техническим обеспечением позволяет существенно снизить влияние негативных факторов и сохранить здоровье пользователей ПЭВМ.

					ПГУ 1.090501.15.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		70

Заключение

В ходе выполнения проекта была разработана автоматизированная информационная система для анализа товарных данных, направленная на оптимизацию процессов учета, оценки и прогнозирования товарных показателей. Созданное решение обеспечивает сбор, хранение, обработку и визуализацию информации о товарах, что значительно упрощает принятие управленческих решений и повышает эффективность бизнеса.

Разрабатываемая система ориентирована на однопользовательский режим работы, что позволяет упростить архитектуру программного продукта, минимизировать затраты на поддержку и обеспечить высокую производительность при выполнении локальных аналитических задач. Это делает её особенно удобной для индивидуальных специалистов или малых предприятий, где критически важна простота в эксплуатации и автономность.

В процессе реализации проекта были сформированы и выполнены ключевые этапы: от анализа предметной области и постановки задачи до проектирования, программной реализации, тестирования и сдачи системы. Все работы осуществлялись в соответствии с календарным планом с четким распределением задач между членами проектной команды.

Использование современных технологий и подходов, а также продуманная организация производственного процесса обеспечили достижение всех поставленных целей. Система обладает модульной архитектурой, возможностью масштабирования и легкой адаптацией под индивидуальные требования заказчика. Проект продемонстрировал высокую степень готовности к внедрению и имеет значительный потенциал для дальнейшего развития и коммерциализации.

					ПГУ 1.090501.15.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		71

Список использованных источников

1. Гончарова Е.А., Ушаков Д.В. Анализ контента в социальных сетях как способ выявления общественного мнения // Социальные и гуманитарные науки. Отечественная и зарубежная литература. Сер. 11, Социология. — 2020. — № 6. — С. 45–52. — Режим доступа: <https://cyberleninka.ru/article/n/avtomatizatsiya-sbora-i-analiza-dannyh-v-sotsialnyh-setyah> (дата обращения: 25.05.2025).

2. Кобяков П.В. Контент-анализ страниц активных пользователей социальной сети «ВКонтакте» // Наука и школа. — 2022. — № 1. — С. 69–72. — Режим доступа: <https://cyberleninka.ru/article/n/kontent-analiz-stranits-aktivnyh-polzovateley-sotsialnoy-seti-vkontakte> (дата обращения: 25.05.2025).

3. Алексеев С.А., Громова А.И. Информационно-аналитические платформы для анализа тональности текстов в социальных сетях // Известия ЮФУ. Технические науки. — 2021. — Т. 225, № 3. — С. 211–218. — Режим доступа: <https://ojs.itmo.ru/index.php/CLCO/article/view/1313> (дата обращения: 25.05.2025).

4. Чижик А.В., Мельникова С.А., Захаров В.П. Анализ тональности комментариев в социальных сетях: создание карты эмоций жителей Санкт-Петербурга // Компьютерная лингвистика и интеллектуальные технологии. — 2022. — № 21. — С. 123–130. — Режим доступа: <https://ojs.itmo.ru/index.php/CLCO/article/view/1313> (дата обращения: 25.05.2025).

5. Сметанин А.В. Анализ тональности текстов из социальных сетей на основе методов машинного обучения // Высшая школа экономики. — 2022. — Режим доступа: https://www.hse.ru/data/2022/10/14/1731127090/Сметанин_резюме_RUS.pdf (дата обращения: 25.05.2025).

6. Обзор информационных систем анализа социальных сетей // КиберЛенинка. — Режим доступа: <https://cyberleninka.ru/article/n/obzor-informatsionnyh-sistem-analiza-sotsialnyh-setey> (дата обращения: 25.05.2025).

7. Ньябуту Г., Мони В., Мбугуа С. Обзор архитектуры клиент-серверных моделей // Международный журнал научных исследований и инженерных

тенденций. — 2024. — Т. 10, № 1. — С. 139–143. — Режим доступа: https://ijsret.com/wp-content/uploads/2024/01/IJSRET_V10_issue1_125.pdf (дата обращения: 25.05.2025).

8. Буч Д., Джейкобсон И., Рамбо Г. UML: Руководство пользователя. — Москва: Вильямс, 2004. — 592 с. — Режим доступа: <https://csharpcooking.github.io/theory/UnifiedModelingLanguageUserGuide.pdf> (дата обращения: 25.05.2025).

9. ГОСТ 19.101–77. Единая система программной документации. Виды программ и программных документов. — Введ. 01.01.1979. — М.: Изд-во стандартов, 1977. — 12 с. — Режим доступа: <https://www.swrit.ru/doc/espd/19.101-77.pdf> (дата обращения: 25.05.2025).

10. REST API: для чего нужен и как работает // Yandex Cloud. — Режим доступа: <https://yandex.cloud/ru/docs/glossary/rest-api> (дата обращения: 25.05.2025).

11. FastAPI: что это и зачем нужен: введение в современный веб-фреймворк // Яндекс Практикум. — 2025. — Режим доступа: <https://practicum.yandex.ru/blog/fastapi-cto-eto-i-zachem-nuzhen/> (дата обращения: 25.05.2025).

12. Psycopg – PostgreSQL database adapter for Python. — Режим доступа: <https://www.psycopg.org/docs/> (дата обращения: 25.05.2025).

13. Pandas - Python Data Analysis Library. — Режим доступа: <https://pandas.pydata.org/> (дата обращения: 25.05.2025).

14. Pydantic Validators. — Режим доступа: <https://docs.pydantic.dev/latest/concepts/validators/> (дата обращения: 25.05.2025).

15. TomSchimansky/CustomTkinter: A modern and customizable python UI-library. — Режим доступа: <https://github.com/TomSchimansky/CustomTkinter> (дата обращения: 25.05.2025).

16. Requests: HTTP for Humans™ — Requests 2.32.3 documentation. — Режим доступа: <https://requests.readthedocs.io/> (дата обращения: 25.05.2025).

17. torch.nn — PyTorch 2.7 документация. — Режим доступа: <https://pytorch.org/docs/stable/nn.html> (дата обращения: 25.05.2025).

18. torch.optim — PyTorch 2.7 документация. — Режим доступа: <https://pytorch.org/docs/stable/optim.html> (дата обращения: 25.05.2025).

19. BERT - Hugging Face. — Режим доступа: https://huggingface.co/docs/transformers/en/model_doc/bert (дата обращения: 25.05.2025).

20. torch.utils.data — PyTorch 2.7 документация. — Режим доступа: <https://pytorch.org/docs/stable/data.html> (дата обращения: 25.05.2025).

21. threading — Параллелизм на основе потоков — документация на Python. — Режим доступа: <https://docs.python.org/3/library/threading.html> (дата обращения: 25.05.2025).

22. webbrowser — Удобный контроллер веб-браузера — Python 3.13.3 документация. — Режим доступа: <https://docs.python.org/3/library/webbrowser.html> (дата обращения: 25.05.2025).

23. tqdm/tqdm: Быстрый, расширяемый индикатор выполнения для Python и CLI. — Режим доступа: <https://github.com/tqdm/tqdm> (дата обращения: 25.05.2025).

24. Диаграмма последовательности // Википедия. — Режим доступа: https://ru.wikipedia.org/wiki/Диаграмма_последовательности (дата обращения: 25.05.2025).

25. Проектирование информационных систем [Электронный ресурс] // Studfile.net. — Режим доступа: <https://studfile.net/preview/9477465/page%3A7/> (дата обращения: 25.05.2025).

26. Налоговый кодекс Российской Федерации (часть вторая): федер. закон от 05.08.2000 № 117-ФЗ (ред. от 28.12.2024) // КонсультантПлюс [Электронный ресурс]. — Режим доступа: http://www.consultant.ru/document/cons_doc_LAW_28165/ (дата обращения: 25.05.2025).

27. Панкрухин А.П. Маркетинг: учебник для вузов / А.П. Панкрухин. — 4-е изд., перераб. и доп. — М.: Омега-Л, 2021. — 528 с.

					ПГУ 1.090501.15.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		74

28. ГОСТ 12.0.003-74. Система стандартов безопасности труда. Общие санитарно-гигиенические требования к воздуху рабочей зоны. — М.: Издательство стандартов, 1974. — 20 с.

29. Типовая инструкция по охране труда при работе на персональных электронно-вычислительных машинах (ТОИ Р-45-084-01) : утверждена приказом Минсвязи РФ №162 от 02.07.2001. — Электрон. дан. — Режим доступа: <http://docs.cntd.ru/document/901740099> (дата обращения: 25.05.2025).

					<i>ПГУ 1.090501.15.001 ПЗ</i>	Лист
Изм.	Лист	№ докум.	Подпись	Дата		75

Приложение А

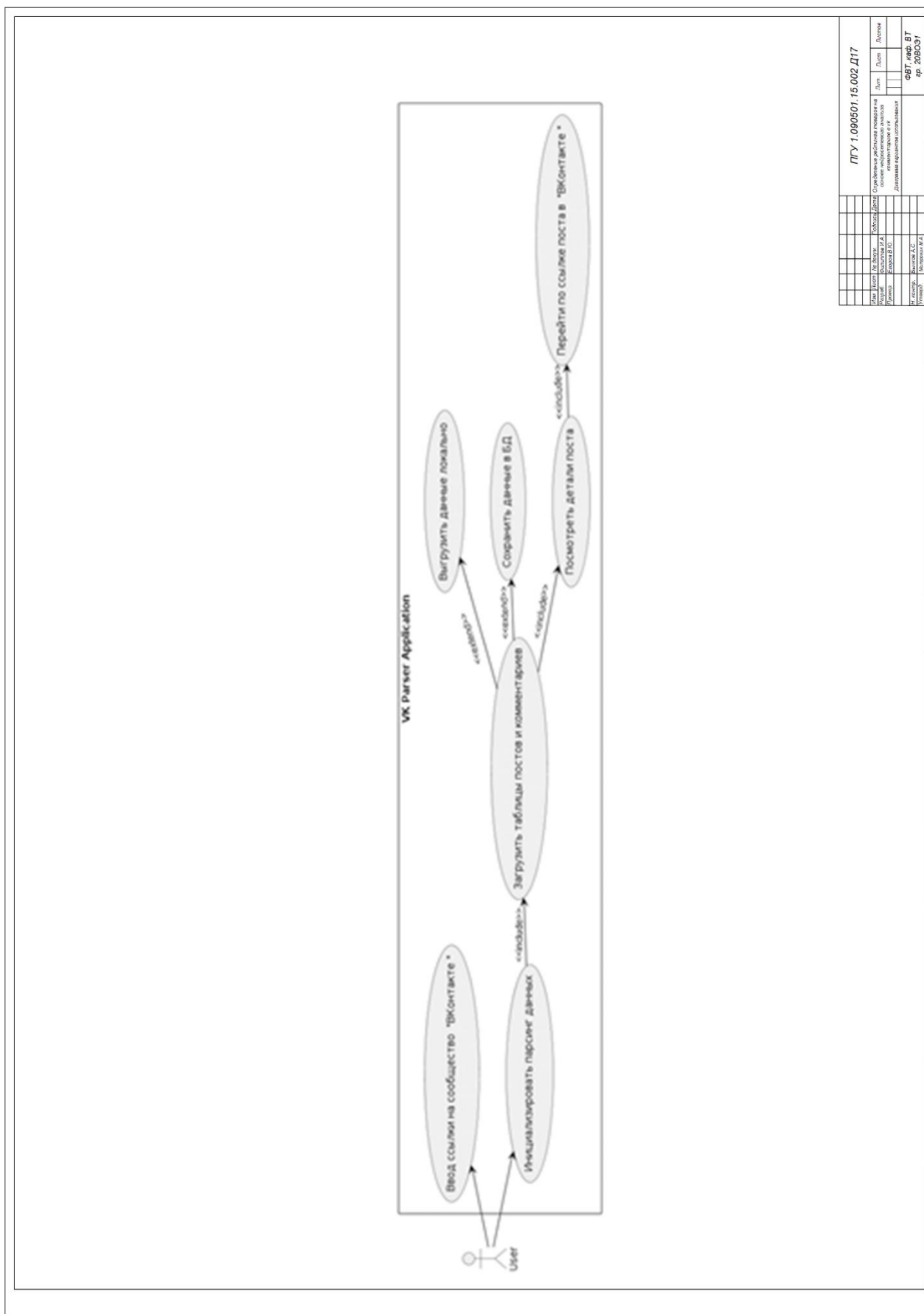
(обязательное)

Программное средство для определение рейтинга товаров на основе
нейросетевого анализа комментариев в vk

UML-диаграммы

					ПГУ 1.090501.15.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		76

Приложение А.1 - Диаграмма вариантов использования



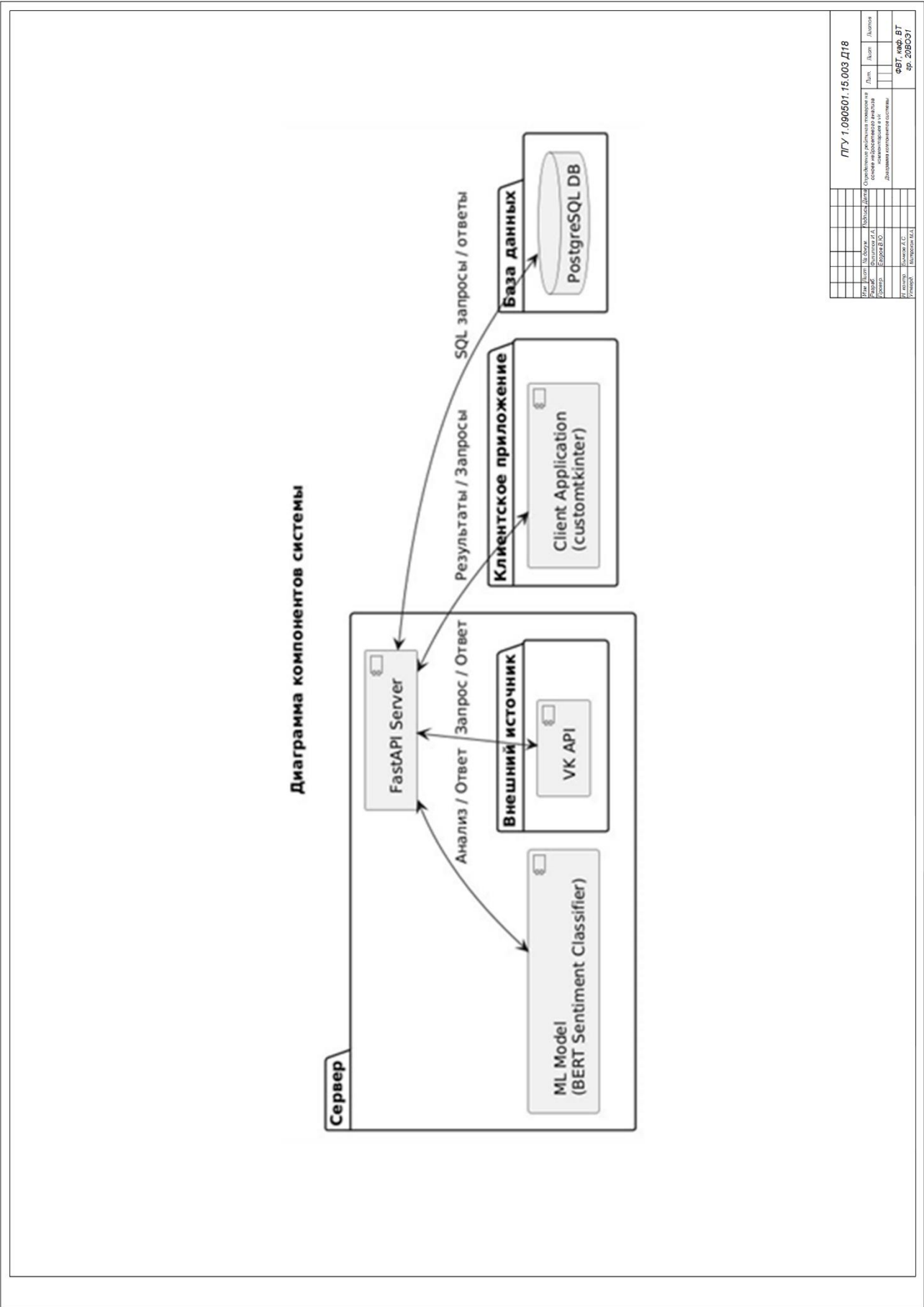
Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090501.15.001 ПЗ

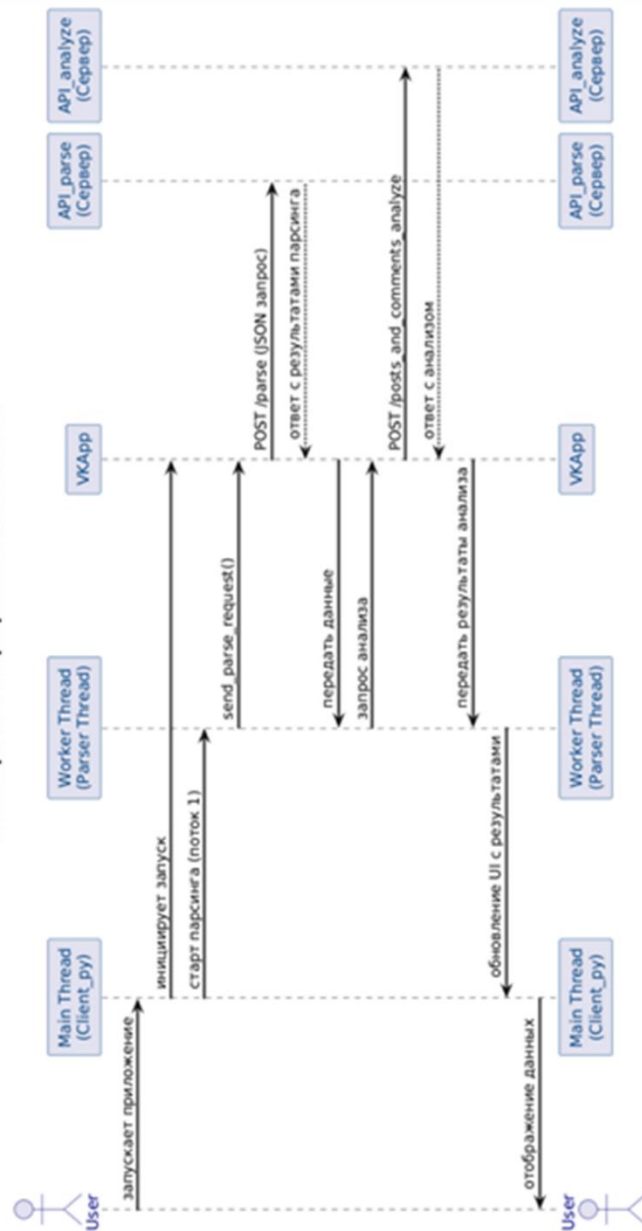
Лист

77

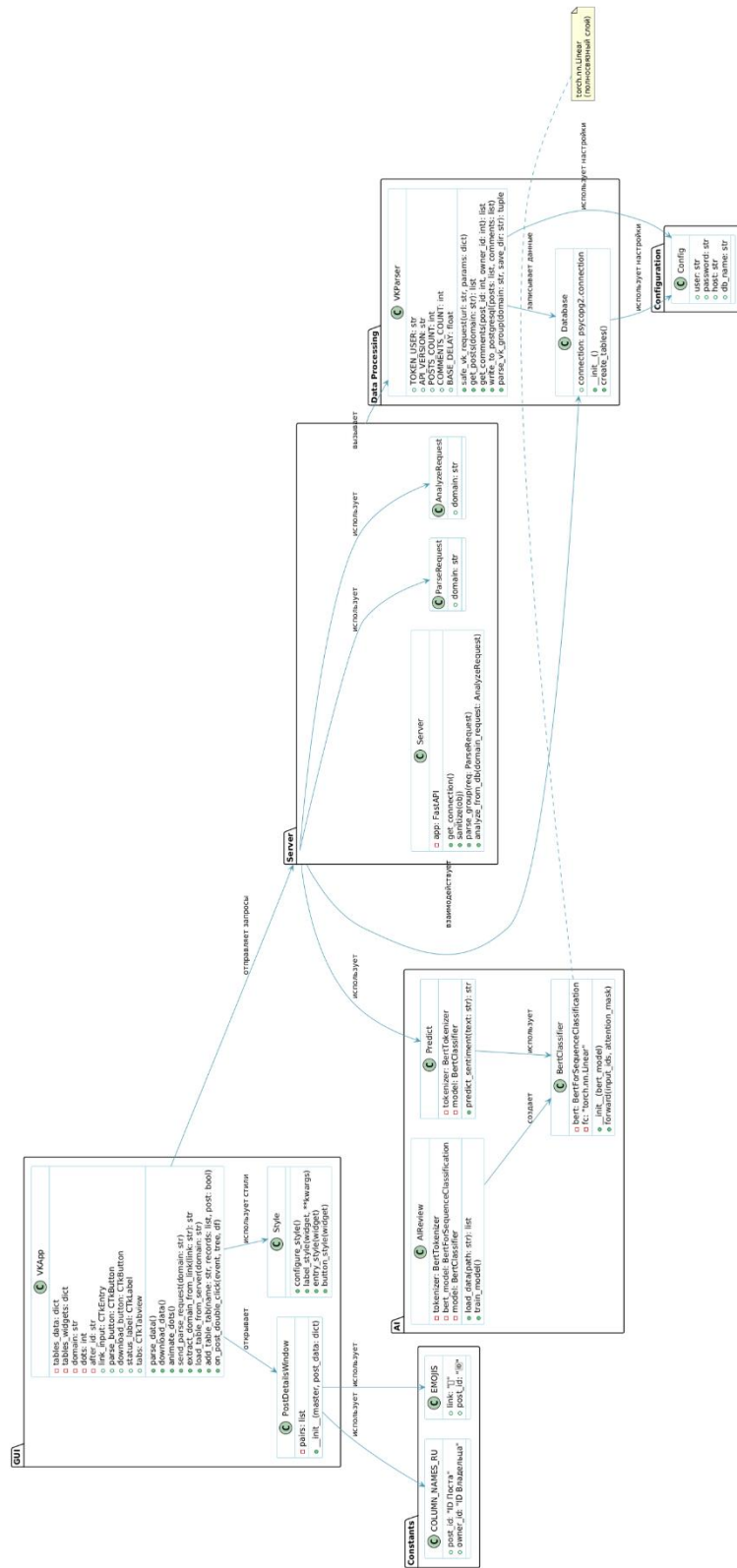
Приложение А.2 - Диаграмма компонентов системы



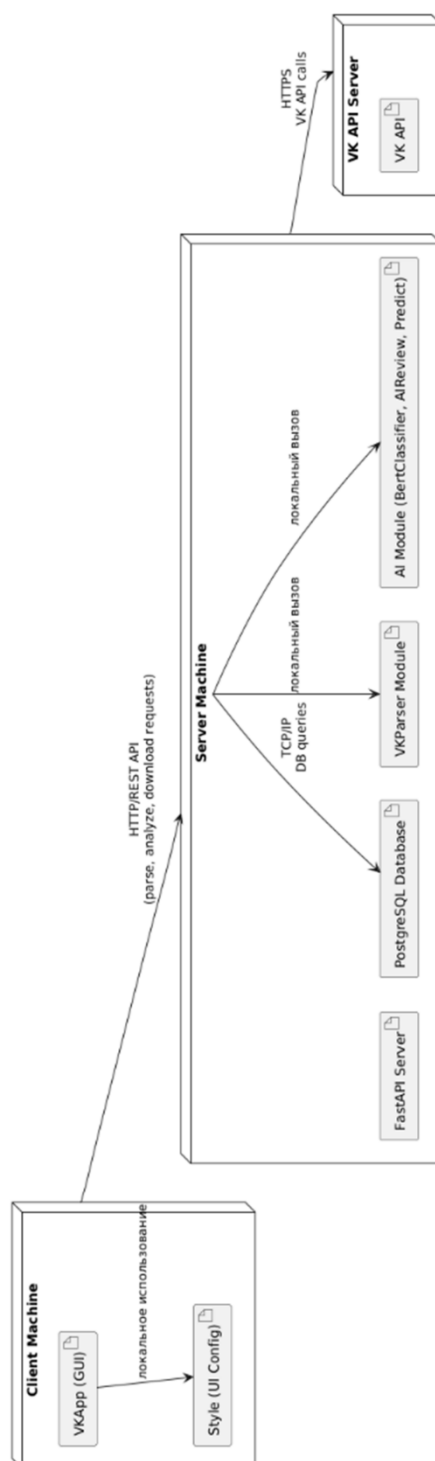
Приложение А.3 - Диаграмма последовательности

[illegible]

Приложение А.4 - Диаграмма классов

[illegible]

Приложение А.5 - Диаграмма разворачивания

[illegible]

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090501.15.001 ПЗ

Лист

81

Приложение Б

(обязательное)

Результаты выполнения тестовых примеров

					ПГУ 1.090501.15.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		82

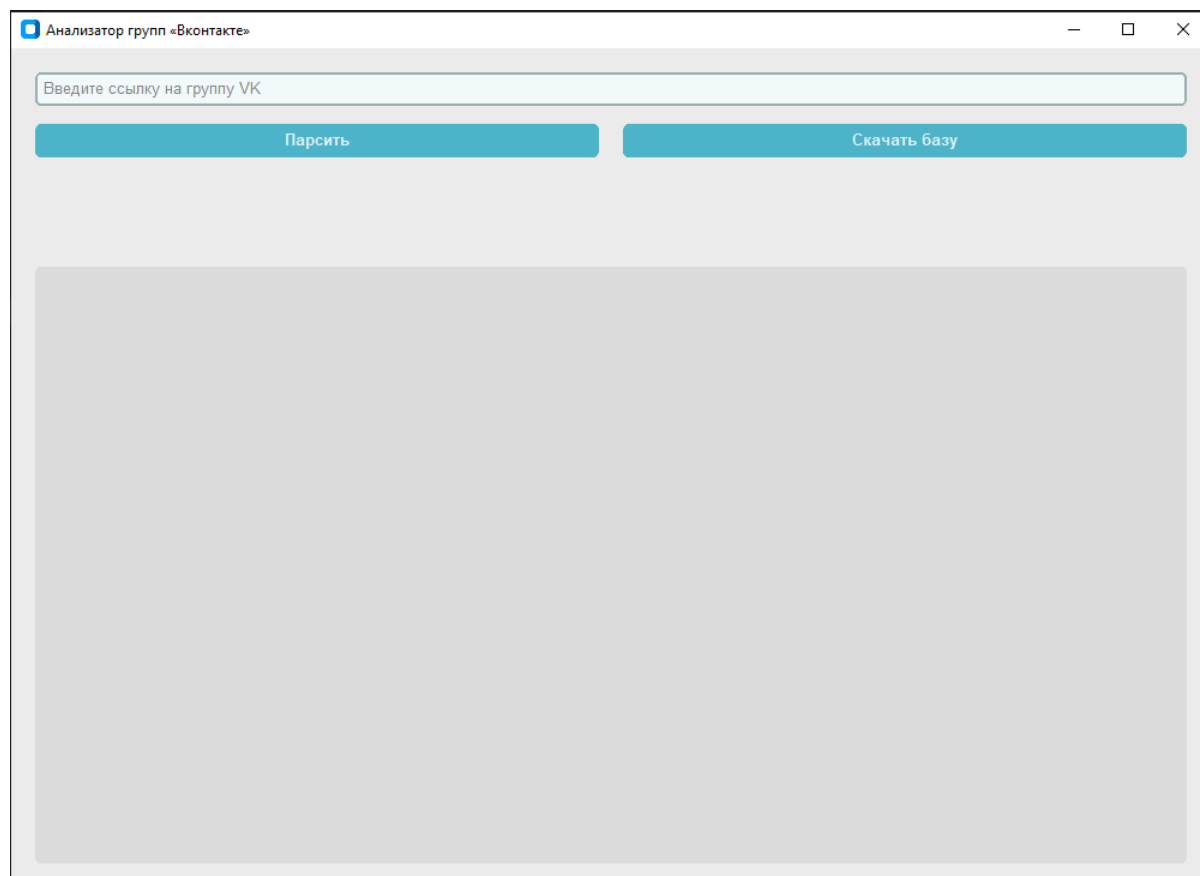


Рисунок Б.1 – Главная страница приложения

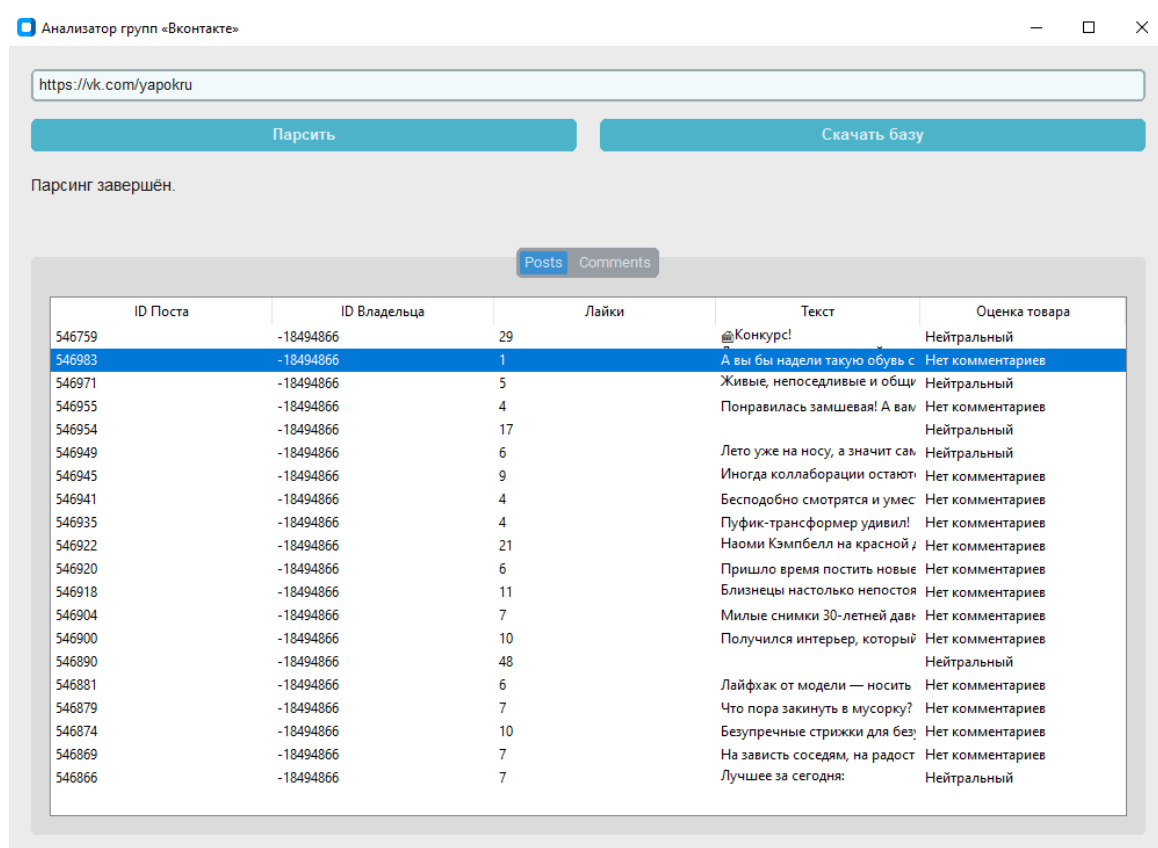


Рисунок Б.2 – Вывод данных сообщества

Ссылка на пост

Открыть пост ВКонтакте

Лайки

4

Текст

Бесподобно смотрятся и уместны всегда!

Оценка товара

Нет комментариев

Рисунок Б.3 – Окно выбранного поста

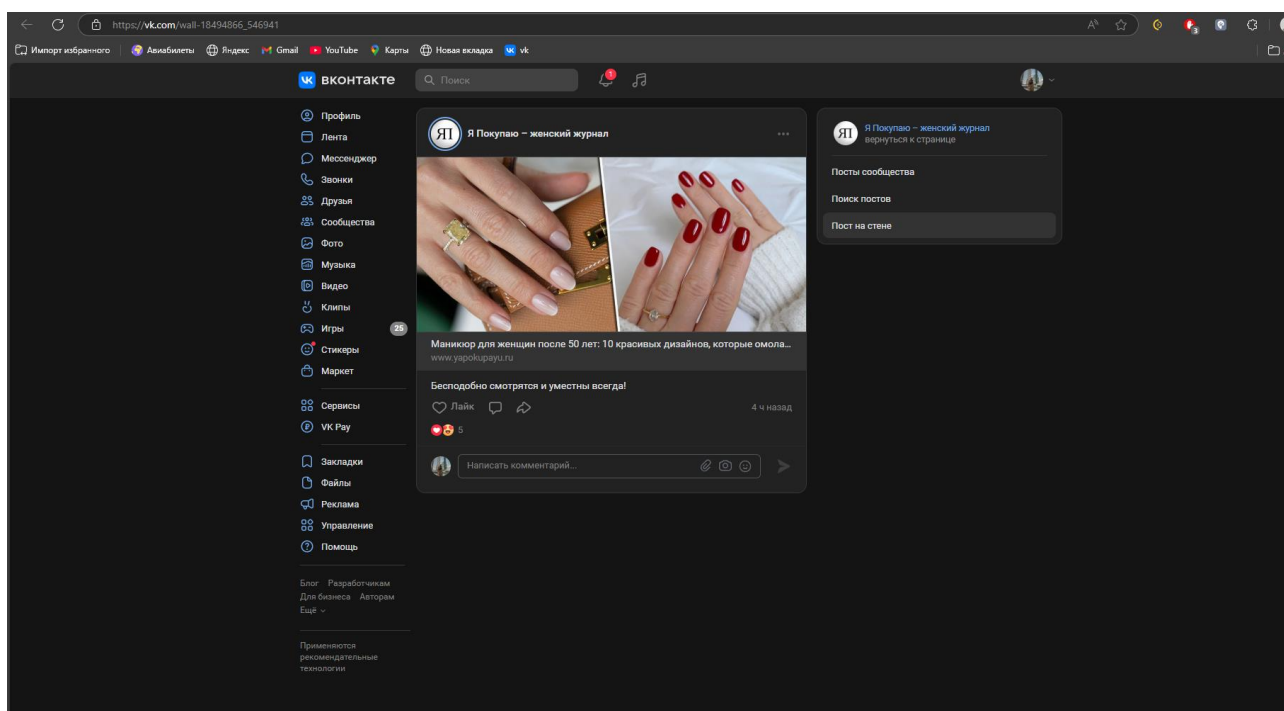


Рисунок Б.4 – Переход на страницу поста «ВКонтакте»

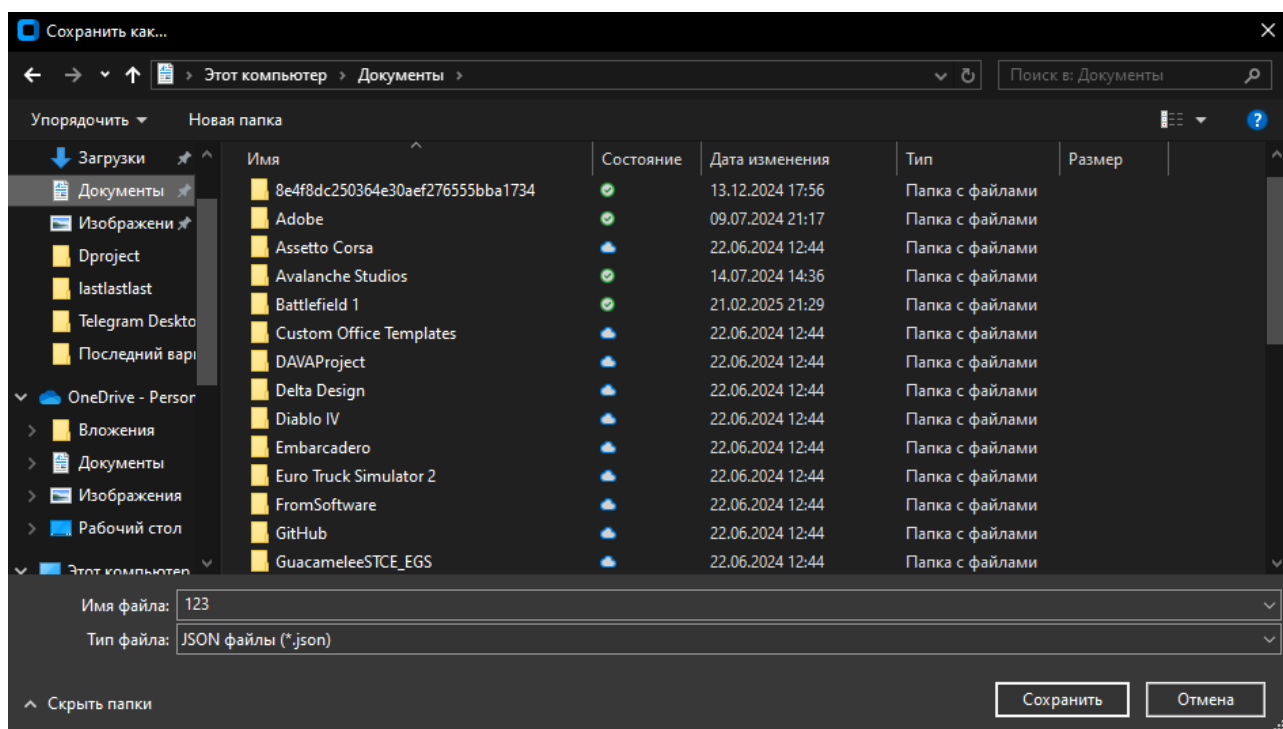


Рисунок Б.5 – Окно сохранения файла

Приложение В
(обязательное)
Листинг программы

					<i>ПГУ 1.090501.15.001 ПЗ</i>	Лист
Изм.	Лист	№ докум.	Подпись	Дата		86

Файл client.py

```
from URLForm import VKApp

if __name__ == "__main__":
    app = VKApp()
    app.mainloop()
```

Файл URLForm.py

```
import json
import requests
import pandas as pd
import os
import re
import threading
import tkinter as tk
import customtkinter as ctk
from tkinter import ttk
from tkinter import messagebox
import webbrowser
import pywintypes
import openpyxl
import webbrowser
import time
from tkinter.filedialog import asksaveasfilename

from style import configure_style, label_style, entry_style, button_style

configure_style()

COLUMN_NAMES_RU = {
    "post_id": "ID Поста",
    "owner_id": "ID Владельца",
    "likes": "Лайки",
    "text": "Текст",
    "comment_id": "ID Комментария",
    "author": "Автор"
}

EMOJIS = {
    "link": "🔗",
    "post_id": "ID",
    "owner_id": "👤",
    "text": "📝",
    "likes": "❤️",
    "Оценка товара": "📊"
}

class VKApp(ctk.CTk):
    def __init__(self):
        super().__init__()

        pywintypes.apply_style(VKApp, style='light')
        self.title("Анализатор групп «Вконтакте»")
```

					ПГУ 1.090501.15.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		87

```

self.geometry("1000x700")
self.minsize(600, 600)

self.tables_data = {}
self.tables_widgets = {}
self.domain = None
self.dots = 0
self.tabs = None
self.after_id = None

self.grid_columnconfigure(0, weight=1)
self.grid_columnconfigure(1, weight=1)

# Ввод ссылки
self.link_input = ctk.CTkEntry(self, placeholder_text="Введите
ссылку на группу VK")
entry_style(self.link_input)
self.link_input.grid(row=0, column=0, columnspan=2, padx=20,
pady=(20, 10), sticky="ew")

# Кнопки
self.parse_button = ctk.CTkButton(self, text="Парсить",
command=self.parse_data)
button_style(self.parse_button)
self.parse_button.grid(row=1, column=0, padx=(20, 10), pady=5,
sticky="ew")

self.download_button = ctk.CTkButton(self, text="Скачать базу",
command=self.download_data)
button_style(self.download_button)
self.download_button.grid(row=1, column=1, padx=(10, 20), pady=5,
sticky="ew")

# Статус
self.status_label = ctk.CTkLabel(self, text="")
label_style(self.status_label)
self.status_label.grid(row=2, column=0, columnspan=2, padx=20,
pady=10, sticky="nsew")

# Вкладки
self.tabs = ctk.CTkTabview(self, width=950, height=500)
self.tabs.grid(row=3, column=0, columnspan=2, padx=20, pady=20,
sticky="nsew")

self.grid_rowconfigure(3, weight=1)

def parse_data(self):
    link = self.link_input.get().strip()
    if not link:
        self.status_label.configure(text="Введите ссылку на группу
VK!")
        return

    domain = self.extract_domain_from_link(link)
    if not domain:
        self.status_label.configure(text="Неверная ссылка.")
        return

```

```

        self.domain = domain
        self.dots = 0
        self.animate_dots()

        thread = threading.Thread(target=self.send_parse_request,
args=(domain,))
        thread.start()

    def download_data(self):
        if not self.tables_data:
            messagebox.showwarning("Нет данных", "Нет данных для
сохранения.")
            return

        file_path = asksaveasfilename(
            defaultextension=".json",
            filetypes=[
                ("JSON файлы", "*.json"),
                ("Excel файлы", "*.xlsx"),
            ],
            title="Сохранить как..."
        )

        if not file_path:
            return

        try:
            if file_path.endswith(".json"):
                data_to_save = {
                    tab_name: df.to_dict(orient="records")
                    for tab_name, df in self.tables_data.items()
                }
                with open(file_path, "w", encoding="utf-8") as f:
                    json.dump(data_to_save, f, ensure_ascii=False,
indent=4)

            elif file_path.endswith(".xlsx"):
                with pd.ExcelWriter(file_path, engine="openpyxl") as
writer:
                    for tab_name, df in self.tables_data.items():
                        df.to_excel(writer, sheet_name=tab_name[:31],
index=False)

            else:
                messagebox.showwarning("Ошибка", "Формат файла не
поддерживается.")
                return

            messagebox.showinfo("Успех", f"Файл сохранён:\n{file_path}")

        except Exception as e:
            messagebox.showerror("Ошибка", f"Не удалось сохранить
файл:\n{e}")

    def animate_dots(self):

```

					ПГУ 1.090501.15.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		89

```

        self.status_label.configure(text="Идёт парсинг" + "." * (self.dots
% 4))
        self.dots += 1
        self.after_id = self.after(500, self.animate_dots)

    def send_parse_request(self, domain):
        try:
            response = requests.post("http://localhost:8000/parse",
json={"domain": domain}, timeout=300)
            if response.status_code == 200:
                self.after_cancel(self.after_id)
                self.status_label.configure(text="Парсинг завершён.
Загружаю таблицы...")
                self.load_table_from_server(domain)
                self.status_label.configure(text="Парсинг завершён.")
            else:
                self.after_cancel(self.after_id)
                self.status_label.configure(text=f"Ошибка сервера:
{response.status_code} - {response.text}")
        except Exception as e:
            self.after_cancel(self.after_id)
            self.status_label.configure(text=f"Сетевой сбой: {e}")

    def extract_domain_from_link(self, link):
        parts = link.strip().split("/")
        return parts[-1] if parts and parts[-1] else parts[-2] if
len(parts) > 1 else None

    def load_table_from_server(self, domain):
        try:
            response =
requests.post("http://localhost:8000/posts_and_comments_analyze",
json={"domain": domain}, timeout=30)
            if response.status_code != 200:
                self.status_label.configure(text=f"Ошибка сервера:
{response.text}")
            return

            data = response.json()
            posts = data.get("posts", [])
            comments = data.get("comments", [])

            if posts:
                self.add_table_tab("Posts", posts, post=True)

            if not comments:
                self.status_label.configure(text="⚠ Комментари
отключены или отсутствуют.")
            else:
                self.add_table_tab("Comments", comments)

        except Exception as e:
            self.status_label.configure(text=f"Ошибка при загрузке: {e}")

    def add_table_tab(self, name, records, post=False):
        df = pd.DataFrame(records)

```

```

# Сохраняем DataFrame
self.tables_data[name] = df

# Проверка – существует ли вкладка
if name in self.tabs._name_list:
    tab = self.tabs.tab(name)
    tree = self.tables_widgets.get(name)
    if tree:
        tree.delete(*tree.get_children()) # очищаем старые данные
    else:
        tree = ttk.Treeview(tab, show="headings",
columns=list(df.columns))
        tree.pack(expand=True, fill="both", padx=10, pady=10)
        self.tables_widgets[name] = tree
    else:
        tab = self.tabs.add(name)
        tree = ttk.Treeview(tab, show="headings",
columns=list(df.columns))
        tree.pack(expand=True, fill="both", padx=10, pady=10)
        self.tables_widgets[name] = tree

# Заголовки колонок
tree["columns"] = list(df.columns)
for col in df.columns:
    russian = COLUMN_NAMES_RU.get(col, col)
    tree.heading(col, text=russian)
    tree.column(col, width=150, anchor="w")

# Вставка данных
for _, row in df.iterrows():
    values = [str(row[col]) for col in df.columns]
    tree.insert("", "end", values=values)

# Клик по посту
if post:
    tree.bind("<Double-1>", lambda event:
self.on_post_double_click(event, tree, df))

def on_post_double_click(self, event, tree, df):
    selected_item = tree.selection()
    if not selected_item:
        return

    item = tree.item(selected_item[0])
    values = item["values"]
    post_data = {col: val for col, val in zip(df.columns, values)}

    dialog = PostDetailsWindow(self, post_data)
    dialog.grab_set()

class PostDetailsWindow(ctk.CTkToplevel):
    def __init__(self, master, post_data):
        super().__init__(master)
        self.title("🔍 Детали поста")
        self.geometry("700x500")
        self.configure(fg_color="white")

```

```

scrollable_frame = ctk.CTkScrollableFrame(self, width=680,
height=480, fg_color="white")
scrollable_frame.pack(padx=10, pady=10, fill="both", expand=True)

# Контейнер для строк с двумя колонками
self.rows_container = ctk.CTkFrame(scrollable_frame,
fg_color="transparent")
self.rows_container.pack(fill="both", expand=True)

self.rows_container.grid_columnconfigure(0, weight=1)
self.rows_container.grid_columnconfigure(1, weight=1)

# Подготовка данных для пар по 2
self.pairs = []
fields = []

# Добавим ссылку в начало, если есть
if "post_id" in post_data and "owner_id" in post_data:
    fields.append(("link", "Ссылка на пост", "Открыть пост
ВКонтакте", True, None))
# остальные поля
for key, value in post_data.items():
    if key in ["post_id", "owner_id"]:
        continue
    display_key = COLUMN_NAMES_RU.get(key, key)
    fields.append((key, display_key, str(value), False, value))

# Формируем пары по 2 для размещения в строках
for i in range(0, len(fields), 2):
    pair = fields[i:i+2]
    self.pairs.append(pair)

# Рендерим пары строками
row_counter = 0 # вручную ведём счётчик строк

for key, title, content, is_link, raw_value in fields:
    # Спец. обработка поля "text"
    if key == "text":
        block = ctk.CTkFrame(
            master=self.rows_container,
            fg_color="white",
            border_color="#ADD8E6",
            border_width=2,
            corner_radius=12
        )
        block.grid(row=row_counter, column=0, columnspan=2,
sticky="nsew", padx=5, pady=8)
        self.rows_container.grid_rowconfigure(row_counter,
weight=1)

        emoji = EMOJIS.get(key, "📄")
        label_title = ctk.CTkLabel(
            master=block,
            text=f"{emoji} {title}",
            font=("Arial", 14, "bold"),
            anchor="center"

```

```

    )
    label_title.pack(pady=(10, 4), anchor="center")

    label_content = ctk.CTkLabel(
        master=block,
        text=content,
        font=("Arial", 13),
        wraplength=640,
        anchor="center"
    )
    label_content.pack(pady=(0, 10), anchor="center")
    row_counter += 1
    continue

# Спец. обработка оценки
elif title == "Оценка товара":
    block = ctk.CTkFrame(
        master=self.rows_container,
        fg_color="white",
        border_color="#ADD8E6",
        border_width=2,
        corner_radius=12
    )
    block.grid(row=row_counter, column=0, columnspan=2,
sticky="nsew", padx=5, pady=8)
    self.rows_container.grid_rowconfigure(row_counter,
weight=1)

    # Цвет текста оценки
    text_color = "gray"
    if content.lower() == "положительный":
        text_color = "green"
    elif content.lower() == "отрицательный":
        text_color = "red"

    emoji = EMOJIS.get(key, "📊")
    label_title = ctk.CTkLabel(
        master=block,
        text=f"{emoji} {title}",
        font=("Arial", 14, "bold"),
        anchor="center"
    )
    label_title.pack(pady=(10, 4), anchor="center")

    label_content = ctk.CTkLabel(
        master=block,
        text=content,
        font=("Arial", 16, "bold"),
        text_color=text_color,
        anchor="center"
    )
    label_content.pack(pady=(0, 10), anchor="center")
    row_counter += 1
    continue

# Все обычные поля (в 2 колонки)
col = row_counter % 2

```

```

        if col == 0:
            current_row = row_counter // 2
            block = ctk.CTkFrame(
                master=self.rows_container,
                fg_color="white",
                border_color="#ADD8E6",
                border_width=2,
                corner_radius=12
            )
            block.grid(row=current_row, column=col, sticky="nsew", padx=5,
                pady=8)
            self.rows_container.grid_rowconfigure(current_row, weight=1)

            emoji = EMOJIS.get(key, "")
            label_title = ctk.CTkLabel(
                master=block,
                text=f"{emoji} {title}",
                font=("Arial", 14, "bold"),
                anchor="center"
            )
            label_title.pack(pady=(10, 4), anchor="center")

            if is_link:
                post_id = post_data["post_id"]
                owner_id = post_data["owner_id"]
                url = f"https://vk.com/wall{owner_id}_{post_id}"

                label_content = ctk.CTkLabel(
                    master=block,
                    text=content,
                    text_color="blue",
                    cursor="hand2",
                    font=("Arial", 13),
                    anchor="center"
                )
                label_content.bind("<Button-1>", lambda e, url=url:
webbrowser.open(url))
            else:
                label_content = ctk.CTkLabel(
                    master=block,
                    text=content,
                    font=("Arial", 13),
                    wraplength=300,
                    anchor="center"
                )
                label_content.pack(pady=(0, 10), anchor="center")
            row_counter += 1

        # Заставляем колонки растягиваться по ширине окна
        self.rows_container.grid_columnconfigure(0, weight=1)
        self.rows_container.grid_columnconfigure(1, weight=1)

```

Файл style.py

```

import customtkinter as ctk

def configure_style():

```

					ПГУ 1.090501.15.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		94

```

# Устанавливаем светлую тему и синий цветовой профиль (по умолчанию)
ctk.set_appearance_mode("light")
ctk.set_default_color_theme("blue")

def label_style(widget, **kwargs):
    widget.configure(
        font=("Arial", 14),
        text_color="gray10",
        anchor="w",
        **kwargs
    )

def entry_style(widget):
    widget.configure(
        font=("Arial", 13),
        width=400,
        corner_radius=6,
        border_width=2,
        fg_color="#F2F9FA",
        border_color="#9AAEB1",
        text_color="gray10",
        placeholder_text_color="gray52"
    )

def button_style(widget):
    widget.configure(
        font=("Arial", 13, "bold"),
        corner_radius=6,
        fg_color="#4DB3C8",
        hover_color="#4A9AB3",
        border_width=0,
        border_color="#4D7A8D",
        text_color="#E1F2F6",
        text_color_disabled="gray74"
    )

```

Файл server.py

```

from fastapi import FastAPI, HTTPException
from pydantic import BaseModel
from Predict import predict_sentiment
from VKParse import parse_vk_group
import psycpg2
import pandas as pd
import math
from config import host, user, password, db_name

app = FastAPI()

class ParseRequest(BaseModel):
    domain: str

class AnalyzeRequest(BaseModel):
    domain: str

def get_connection():
    return psycpg2.connect(

```

					ПГУ 1.090501.15.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		95

```

        host=host,
        user=user,
        password=password,
        database=db_name
    )
#Чистка JSON
def sanitize(obj):
    if isinstance(obj, float):
        if math.isnan(obj) or math.isinf(obj):
            return None # можно заменить на "NaN", 0 и т.п.
        return obj
    elif isinstance(obj, dict):
        return {k: sanitize(v) for k, v in obj.items()}
    elif isinstance(obj, list):
        return [sanitize(i) for i in obj]
    elif obj is None:
        return ""
    return obj

@app.post("/parse")
def parse_group(req: ParseRequest):
    try:
        # Парсим данные из VK
        posts, comments = parse_vk_group(req.domain)

        # Преобразуем в DataFrame
        df_posts = pd.DataFrame(posts)
        df_posts["likes"] = df_posts["likes"].apply(lambda x:
x.get("count", 0) if isinstance(x, dict) else 0)

        df_comments = pd.DataFrame(comments)
        df_comments["likes"] = df_comments["likes"].apply(lambda x: int(x)
if isinstance(x, int) else 0)

        # Возвращаем в браузер как JSON
        response = {
            "status": "OK",
            "posts": df_posts.to_dict(orient="records"),
            "comments": df_comments.to_dict(orient="records")
        }
        return sanitize(response)

    except Exception as e:
        print(f"Ошибка парсинга/записи: {e}")
        raise HTTPException(status_code=500, detail=f"Ошибка
парсинга/записи: {e}")

@app.post("/posts_and_comments_analyze")
def analyze_from_db(domain_request: AnalyzeRequest):
    try:
        with get_connection() as conn:
            df_posts = pd.read_sql("SELECT * FROM posts", conn)
            df_comments = pd.read_sql("SELECT * FROM comments", conn)

            if df_posts.empty:
                raise HTTPException(status_code=404, detail="Нет постов по
данному домену")

```

```

        if df_comments.empty:
            raise HTTPException(status_code=404, detail="Нет комментариев
по данному домену")

        # Прогноз тональности
        df_comments["Оценка тональности"] =
df_comments["text"].astype(str).apply(predict_sentiment)

        sentiment_map = {"Отрицательный": -1, "Нейтральный": 0,
"Положительный": 1}
        reverse_map = {-1: "Отрицательный", 0: "Нейтральный", 1:
"Положительный"}

        # НЕ сохраняем sentiment_score в df_comments
        sentiment_scores = df_comments["Оценка
тональности"].map(sentiment_map)

        avg_sentiment = (
            df_comments[["post_id"]].copy()
            .assign(score=sentiment_scores)
            .groupby("post_id")["score"]
            .mean()
            .round()
            .astype(int)
            .map(reverse_map)
        )

        df_posts["Оценка товара"] =
df_posts["post_id"].map(avg_sentiment).fillna("Нет комментариев")

        return sanitize({
            "posts": df_posts.fillna("").to_dict(orient="records"),
            "comments": df_comments.fillna("").to_dict(orient="records")
        })

    except Exception as e:
        raise HTTPException(status_code=500, detail=str(e))

```

Файл AIReview.py

```

import torch
import torch.nn as nn
import torch.optim as optim
import pandas as pd
import time
from tqdm import tqdm
from collections import Counter
from transformers import BertTokenizer, BertForSequenceClassification
from torch.utils.data import DataLoader, Dataset

# Загрузка токенизатора и модели BERT
tokenizer = BertTokenizer.from_pretrained("DeepPavlov/rubert-base-cased")
bert_model =
BertForSequenceClassification.from_pretrained("DeepPavlov/rubert-base-
cased", num_labels=3)

# Загрузка данных из CSV

```

```

def load_data(path):
    df = pd.read_excel(path, sheet_name=0)

    if "Russian_Review" not in df.columns or "Lable_Num" not in
df.columns:
        raise ValueError("В файле отсутствуют столбцы 'Russian_Review' и
'Lable_Num'.")

    def label_to_number(label):
        if label == "позитивный":
            return 2
        elif label == "отрицательный":
            return 0
        elif label == "нейтральный":
            return 1
        else:
            return -1 # Возвращаем -1, если метка не соответствует
ожидаемым

    df = df[~df["Russian_Review"].apply(lambda x: isinstance(x, str) and
x.strip() == "#VALUE!")]
    # Применяем функцию к столбцу Lable_Num
    df["Lable_Num"] = df["Lable_Num"].apply(label_to_number)

    # Возвращаем список кортежей (предполагается, что столбцы называются
именно так)
    return list(zip(df["Russian_Review"].tolist(),
df["Lable_Num"].tolist()))

data = load_data("DatasetNew.xlsx")

# Подсчет частоты классов и вычисление весов
class_counts = Counter([label for _, label in data])
class_weights = torch.tensor([1.0 / class_counts[i] for i in range(3)],
dtype=torch.float)

# Создание кастомного Dataset
class TextDataset(Dataset):
    def __init__(self, data):
        self.data = data

    def __len__(self):
        return len(self.data)

    def __getitem__(self, idx):
        text, label = self.data[idx]
        encoding = tokenizer(text, padding="max_length", truncation=True,
max_length=32, return_tensors="pt")
        return encoding["input_ids"].squeeze(0),
encoding["attention_mask"].squeeze(0), torch.tensor(label)

# Создание DataLoader
dataset = TextDataset(data)
dataloader = DataLoader(dataset, batch_size=8, shuffle=True)

# Определение модели классификации
class BertClassifier(nn.Module):

```

```

def __init__(self, bert_model):
    super(BertClassifier, self).__init__()
    self.bert = bert_model
    self.fc = nn.Linear(768, 3)  # 3 класса (2,1,0)

    def forward(self, input_ids, attention_mask):
        outputs = self.bert(input_ids=input_ids,
        attention_mask=attention_mask)
        logits = outputs.logits  # Получаем логиты
        return logits

# Инициализация модели, функции потерь и оптимизатора
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = BertClassifier(bert_model).to(device)
class_weights = class_weights.to(device)
criterion = nn.CrossEntropyLoss(weight=class_weights)
optimizer = optim.Adam(model.parameters(), lr=1e-5)

# Обучение модели
epochs = 30
for epoch in range(epochs):
    total_loss = 0

    start_time = time.time()  # Засекаем время начала эпохи
    progress_bar = tqdm(dataloader, desc=f"Epoch {epoch+1}", leave=True)

    for input_ids, attention_mask, labels in dataloader:
        input_ids, attention_mask, labels = input_ids.to(device),
        attention_mask.to(device), labels.to(device)

        optimizer.zero_grad()
        outputs = model(input_ids, attention_mask)
        loss = criterion(outputs, labels)
        loss.backward()
        torch.nn.utils.clip_grad_norm_(model.parameters(),
        max_norm=1.0)  # Gradient Clipping
        optimizer.step()

        total_loss += loss.item()
        progress_bar.set_postfix(loss=loss.item())
        progress_bar.update(1)

    progress_bar.close()

    end_time = time.time()  # Засекаем конец эпохи
    epoch_time = end_time - start_time  # Вычисляем длительность эпохи

    print(f"Epoch {epoch+1} completed: Loss={total_loss /
    len(dataloader):.4f}, Time={epoch_time:.2f}s,
    Speed={len(dataloader)/epoch_time:.2f} batches/sec")

model_save_path = "bert_classifier.pth"
torch.save(model.state_dict(), model_save_path)
print(f"Training complete! Model saved to {model_save_path}")

```

Файл config.py

					ПГУ 1.090501.15.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		99

```

user = "postgres"
password = "qwerty123if"
host = "localhost"
db_name = "vkparser"

```

Файл database.py

```

import psycopg2
from config import host, user, password, db_name

try:
    connection = psycopg2.connect(
        host=host,
        user=user,
        password=password,
        database=db_name
    )

    connection.autocommit = True

    with connection.cursor() as cursor:
        cursor.execute(
            "SELECT version();"
        )

        print(f"Server version: {cursor.fetchone()}")

    with connection.cursor() as cursor:
        # Таблица постов
        cursor.execute("""
            CREATE TABLE IF NOT EXISTS posts (
                post_id INTEGER PRIMARY KEY,
                owner_id INTEGER NOT NULL,
                likes INTEGER NOT NULL DEFAULT 0,
                text TEXT NOT NULL
            );
        """)

        # Таблица комментариев
        cursor.execute("""
            CREATE TABLE IF NOT EXISTS comments (
                comment_id INTEGER PRIMARY KEY,
                post_id INTEGER NOT NULL REFERENCES posts(post_id) ON
DELETE CASCADE,
                author TEXT NOT NULL,
                text TEXT NOT NULL,
                likes INTEGER NOT NULL DEFAULT 0
            );
        """)

        print("Таблицы успешно созданы или обновлены")

except Exception as _ex:
    print("[INFO] Error while working with PostgreSQL")

```

					ПГУ 1.090501.15.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		100

```
finally:
    if connection:
        connection.close()
        print("[INFO] PostgreSQL connection close")
```

Файл Predict.py

```
import torch
import torch.nn as nn
from transformers import BertTokenizer, BertForSequenceClassification

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
tokenizer = BertTokenizer.from_pretrained("DeepPavlov/rubert-base-cased")

class BertClassifier(nn.Module):
    def __init__(self, bert_model):
        super(BertClassifier, self).__init__()
        self.bert = bert_model
        self.fc = nn.Linear(768, 3) # 3 класса (позитивный, нейтральный, отрицательный)

    def forward(self, input_ids, attention_mask):
        outputs = self.bert(input_ids=input_ids,
                             attention_mask=attention_mask)
        logits = outputs.logits # Получаем логиты
        return logits

# Загрузка модели
bert_model =
BertForSequenceClassification.from_pretrained("DeepPavlov/rubert-base-
cased", num_labels=3)
model = BertClassifier(bert_model).to(device)
model.load_state_dict(torch.load("bert_classifier.pth",
map_location=device))
model.eval()

sentiment_labels = ["Отрицательный", "Нейтральный", "Положительный"]

def predict_sentiment(text):
    encoding = tokenizer(text, padding="max_length", truncation=True,
max_length=32, return_tensors="pt")
    input_ids = encoding["input_ids"].to(device)
    attention_mask = encoding["attention_mask"].to(device)

    with torch.no_grad():
        outputs = model(input_ids, attention_mask)
        _, preds = torch.max(outputs, dim=1)

    return sentiment_labels[preds.item()]
```

Файл VKParse.py

```
import sys
import time
import os
import requests
```

					ПГУ 1.090501.15.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		101

```

import pandas as pd
from random import uniform
import psycopg2
from config import host, user, password, db_name

# Константы
TOKEN_USER =
'040376c7040376c7040376c748072f289100403040376c763d3027952e961f92d7ca0fc'
API_VERSION = '5.199'
POSTS_COUNT = 20
COMMENTS_COUNT = 20
BASE_DELAY = 0.4

def safe_vk_request(url, params):
    """Защищённый запрос с анти-спамом и ретраем"""
    while True:
        time.sleep(BASE_DELAY + uniform(0.2, 0.4))
        try:
            response = requests.get(url, params=params)
            data = response.json()
        except Exception as e:
            print(f"Сетевая ошибка: {e}")
            continue

        if 'error' in data:
            if data['error']['error_code'] == 6:
                print("⚠ Слишком много запросов. Пауза 1.5 секунды...")
                time.sleep(1.5)
                continue
            else:
                print(f"❌ Ошибка VK API: {data['error']['error_msg']}")
                return None
        return data

def get_posts(domain):
    offset = 0
    all_posts = []

    while offset < POSTS_COUNT:
        params = {
            'access_token': TOKEN_USER,
            'v': API_VERSION,
            'domain': domain,
            'count': POSTS_COUNT,
            'offset': offset
        }

        data = safe_vk_request('https://api.vk.com/method/wall.get',
params)
        if data is None:
            break

        items = data.get('response', {}).get('items', [])
        if not items:
            break

        all_posts.extend(items)

```

```

        offset += POSTS_COUNT

    return all_posts

def get_comments(post_id, owner_id):
    offset = 0
    all_comments = []

    while offset < COMMENTS_COUNT:
        params = {
            'access_token': TOKEN_USER,
            'v': API_VERSION,
            'owner_id': owner_id,
            'post_id': post_id,
            'count': COMMENTS_COUNT,
            'offset': offset,
            'preview_length': 0,
            'extended': 1,
            'fields': 'first_name,last_name'
        }

        try:
            data =
safe_vk_request('https://api.vk.com/method/wall.getComments', params)
        except Exception as e:
            print(f"Ошибка запроса комментариев: {e}")
            break

        if 'error' in data:
            error_msg = data['error'].get('error_msg', 'Неизвестная
ошибка')
            print(f"Ошибка при получении комментариев к посту {post_id}:
{error_msg}")
            # Проверяем, была ли ошибка из-за отключенных комментариев
            if 'wall.comments.disabled' in error_msg:
                print(f"Комментарии отключены для поста {post_id}.")
                break

            response_data = data.get('response', {})
            comments = response_data.get('items', [])
            profiles = response_data.get('profiles', [])

            # Если комментарии отключены, выводим сообщение и завершаем
            обработку.
            if not comments:
                print(f"Комментарии отключены для поста {post_id}.")
                break

            user_map = {
                user['id']: f"{user['first_name']} {user['last_name']}"
                for user in profiles
            }

            for comment in comments:
                user_id = comment.get('from_id', 'Unknown')
                author = user_map.get(user_id, f"User {user_id}")

```

```

        all_comments.append({
            'post_id': post_id,
            'comment_id': comment.get('id'),
            'author': author,
            'text': comment.get('text', ''),
            'likes': comment.get('likes', {}).get('count', 0)
        })

    offset += COMMENTS_COUNT

    return all_comments

def write_to_postgresql(posts, comments):
    try:
        conn = psycopg2.connect(
            host=host,
            user=user,
            password=password,
            database=db_name
        )
        conn.autocommit = True
        cursor = conn.cursor()

        cursor.execute("DELETE FROM comments;")
        cursor.execute("DELETE FROM posts;")

        for post in posts:
            cursor.execute(
                """
                INSERT INTO posts (post_id, owner_id, likes, text)
                VALUES (%s, %s, %s, %s)
                ON CONFLICT (post_id) DO NOTHING;
                """,
                (
                    post.get('id'),
                    post.get('owner_id'),
                    post.get('likes', {}).get('count', 0),
                    post.get('text', ''),
                )
            )

        for comment in comments:
            cursor.execute(
                """
                INSERT INTO comments (comment_id, post_id, author, text,
likes)
                VALUES (%s, %s, %s, %s, %s)
                ON CONFLICT (comment_id) DO NOTHING;
                """,
                (
                    comment.get('comment_id'),
                    comment.get('post_id'),
                    comment.get('author', 'anonymous'),
                    comment.get('text', ''),
                    comment.get('likes', 0)
                )
            )
    
```

```

        print("Данные записаны в PostgreSQL")
        cursor.close()
        conn.close()

    except Exception as e:
        print(f"Ошибка записи в PostgreSQL: {e}")

def parse_vk_group(domain, save_dir="."):
    posts = get_posts(domain)
    comments = []

    for post in posts:
        post_comments = get_comments(post['id'], post['owner_id'])
        if post_comments:
            comments.extend(post_comments)

    write_to_postgresql(posts or [], comments or [])

    return posts or [], comments or []

# CLI
if __name__ == '__main__':
    if len(sys.argv) > 1:
        parse_vk_group(sys.argv[1])
    else:
        sys.exit(1)

```

Приложение Г

(Обязательное)

Программное средство для определение рейтинга товаров на основе
нейросетевого анализа комментариев в vk

Презентация

					ПГУ 1.090501.15.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		106

ОПРЕДЕЛЕНИЕ РЕЙТИНГА ТОВАРОВ НА ОСНОВЕ НЕЙРОСЕТЕВОГО АНАЛИЗА КОММЕНТАРИЕВ В VK

Выполнил студент группы
20ВОЭ1: Филиппов И.А.

Научный руководитель:
к.т.н., доцент,
Егоров В.Ю.

ЦЕЛЬ РАБОТЫ И ЗАДАЧИ

Цель работы – разработка программного средства, определяющего рейтинг товаров на основе нейросетевого анализа

Задачи:

- Парсинг данных из сообщества «Вконтакте»
- Осуществлять связь с сервером при помощи http-запросов
- Производить анализ товаров
- Просматривать детали поста
- Сохранять данные в удобных форматах (JSON, EXCEL)

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090501.15.001 ПЗ

Лист

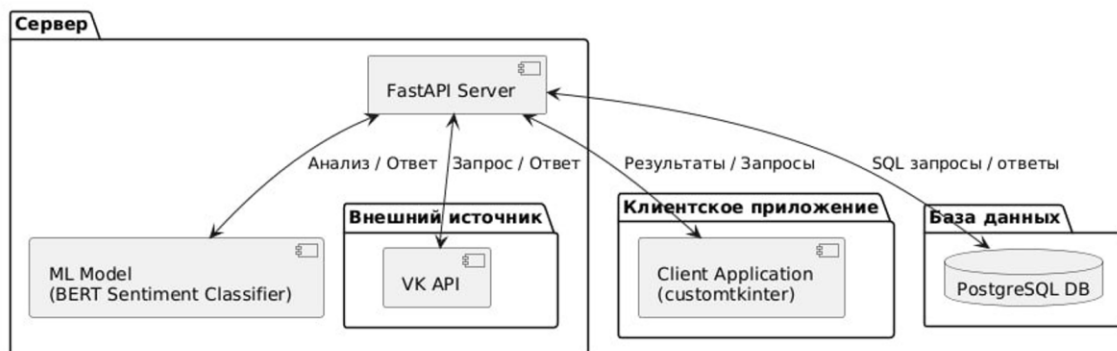
107

АКТУАЛЬНОСТЬ ТЕМЫ

Актуальность данной темы обусловлена стремительным ростом социальных сетей и онлайн-платформ, что привело к резкому увеличению объёма пользовательских отзывов и комментариев. Это создало необходимость в эффективных методах автоматизированного анализа большого массива текстовой информации для своевременного и точного выявления настроений и мнений.

АРХИТЕКТУРА КОМПОНЕНТОВ СИСТЕМЫ

Диаграмма компонентов системы



ТЕХНОЛОГИЧЕСКИЙ СТЕК

Клиентская часть

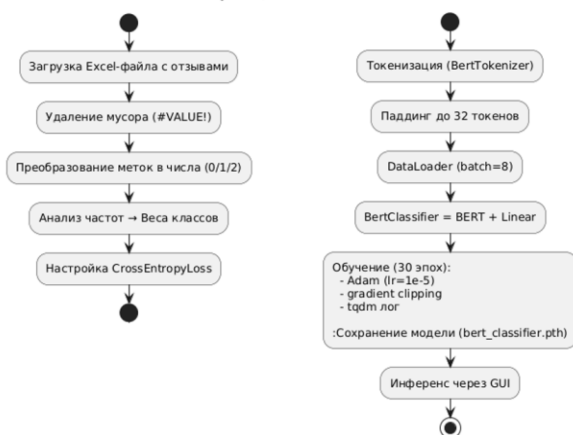
- Python 1.13.2
- CustomTkinter GUI
- Windows + Linux

Серверная часть

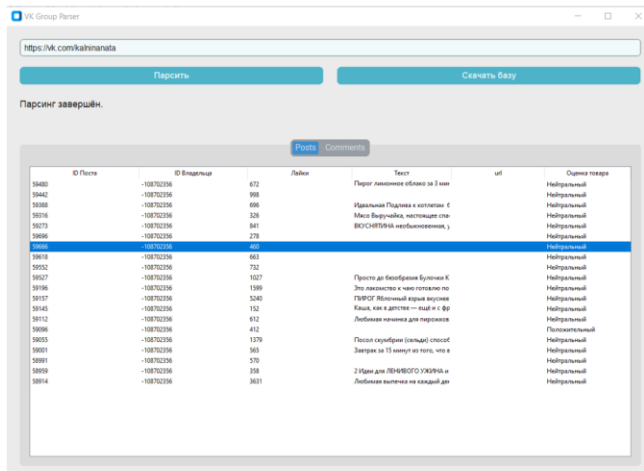
- Python 1.13.2
- API: VK API, FastAPI
- ML: PyTorch + transformers
- DB: PostgreSQL + psycorg2

Модуль машинного обучения

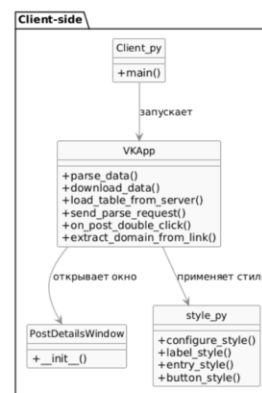
Классификация отзывов с BERT



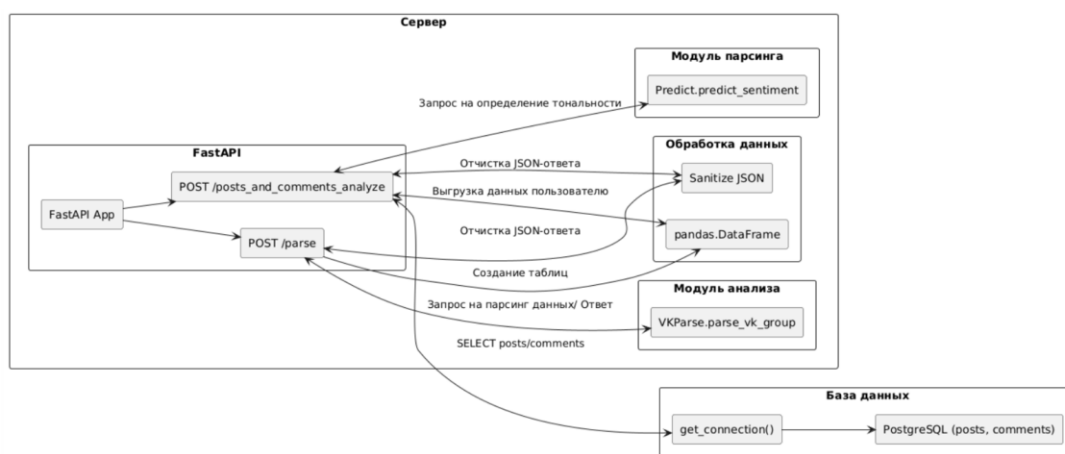
Клиентская часть



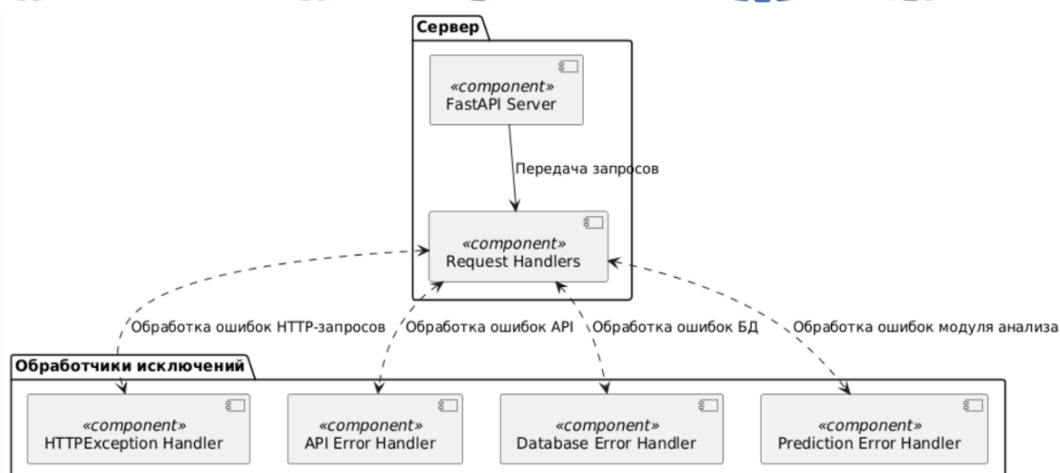
Архитектура клиентской части VK Group Parser



Архитектура серверной части системы



Модель обработки ошибок



Демонстрация работы

VK Group Parser

https://vk.com/kaitanata

Парсить Сканать базу

Парсинг завершен.

ID Поста	ID Владельца	Лайки	Текст	url	Оценка товара
59480	-108762336	672	Паркет ламинированный 3 мм	Найденный	Найденный
59482	-108762336	668	Идеальная Подушка в комплекте €	Найденный	Найденный
59486	-108762336	686	Миски Ветеринары, настоящие оле	Найденный	Найденный
59476	-108762336	841	800 Сабвуфер (высококачественный, з	Найденный	Найденный
59496	-108762336	278	Найденный	Найденный	Найденный
59478	-108762336	683	Найденный	Найденный	Найденный
59532	-108762336	712	Просто до баночки Булочки К	Найденный	Найденный
59527	-108762336	1627	Это ламинированный и очень красивый	Найденный	Найденный
59196	-108762336	1099	ПВХ/МДФ/Массивный экран зеркала	Найденный	Найденный
59157	-108762336	5240	Каша, как и детские — мой и с ф	Найденный	Найденный
59145	-108762336	152	Любимая кошечка для переезда	Найденный	Найденный
59112	-108762336	612	Найденный	Найденный	Найденный
59096	-108762336	412	Найденный	Найденный	Найденный
59055	-108762336	1379	После службы (сладкий стейк)	Найденный	Найденный
59001	-108762336	385	Защита от 15 минут из того, что в	Найденный	Найденный
58981	-108762336	270	2 Идет для ЛЕНОВОГО УЖИНА и	Найденный	Найденный
58959	-108762336	358	Найденный	Найденный	Найденный
58914	-108762336	1621	Найденный	Найденный	Найденный

Детали поста

Ссылка на пост
Открыть пост ВКонтакте

Лайки
4

Текст
Беспокойно смотреть и уместны всегда!

Оценка товара
Нет комментариев

СПАСИБО ЗА ВНИМАНИЕ!



					ПГУ 1.090501.15.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		112