

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
вычислительной техники

Кафедра
МОиПЭВМ

Направление подготовки 09.03.04 «Программная инженерия»

БАКАЛАВРСКАЯ РАБОТА

на тему

**Программная реализация аналитической системы
мониторинга Telegram-каналов**

Студент

(подпись, дата)

Мартынов Илья Сергеевич

(ФИО полностью)

Руководитель

(подпись, дата)

Афонин А.Ю.

(фамилия, инициалы)

Нормоконтролер

(подпись, дата)

Такташкин Д.В.

(фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от _____ № _____)

Заведующий кафедрой

(подпись)

Козлов А.Ю.

(фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от _____ № _____)

Секретарь ГЭК

(подпись)

Попова Н.А.

(фамилия, инициалы)

Пенза, 2023

Федеральное государственное бюджетное образовательное учреждение
высшего образования

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет

Вычислительной техники

Кафедра

МО и ПЭВМ

«Утверждаю»

Заведующий кафедрой

_____ А.Ю. Козлов

«__» _____ 2023 г

ЗАДАНИЕ

НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

БАКАЛАВРА

1. Студент Мартынов Илья Сергеевич гр. 19ВП2 факультета

<Фамилия, имя, отчество полностью>

вычислительной техники

2. Тема работы Программная реализация аналитической системы мониторинга Telegram-каналов

Тема утверждена приказом ПГУ № _____ от " ____ " апреля 2023 года

2. Руководитель работы Афонин Александр Юрьевич

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

1) *Назначение: система анализа и мониторинга Telegram каналов*

2) *Среда выполнения: операционная система Windows 10*

Языковые средства разработки: PHP, JavaScript, HTML5, CSS3.

Инструментальные средства: PhpStorm 2023, Draw.io.

Технологии разработки: ООП, принципы SOLID, шаблоны проектирования, клиент-серверная архитектура API.

3) *Основные функции приложения:*

а) анализ источников информации;

б) просмотр публикаций по интересующим темам;

в) формирование статистики Telegram-канала;

4. Перечень подлежащих разработке вопросов:

1) Анализ предметной области и постановка задачи;

2) Проектирование;

3) Реализация;

4) Контроль качества программного обеспечения;

5) План разработки бюджета;

6) Оформление пояснительной записки.

5. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и требований к системе	10%	12.04.2023 – 16.04.2023	
2	Проектирование программных средств	20%	17.04.2023 – 26.04.2023	
3	Реализация программных средств	30%	27.04.2023 – 11.05.2023	
4	Тестирование программных средств	20%	12.05.2023 – 21.05.2023	
5	Оформление пояснительной записки	20%	22.05.2023 – 01.06.2023	

Дата выдачи задания «22» декабря 2022г.

Руководитель бакалаврской работы _____
(подпись, дата)

Задание к исполнению принял студент _____
(подпись, дата)

Работу к защите допустить

Декан ФВТ _____ Л.Р. Фионова
(подпись, дата)

Реферат

Пояснительная записка содержит 81 лист, 23 рисунка, 9 таблиц, 12 использованных источников и 2 приложения.

TELEGRAM, КЛИЕНТ, СЕРВЕР, ВЕБ-ИНТЕРФЕЙС, UML.

Объект исследования – средства анализа и обобщения информации, получаемой из Telegram-каналов.

Цель выпускной квалификационной работы – создание эффективной, надежной и простой в использовании аналитической системы мониторинга, позволяющую пользователям обобщать и анализировать информацию, размещаемую в Telegram-каналах.

Технологии разработки – разработка приложения проводилась на языке программирования PHP в среде PHPStorm 2023. Веб-интерфейс разработан на языке HTML и CSS.

Среда исполнения – веб-браузер.

Результаты работы – в ходе выполнения выпускной квалификационной работы были созданы клиентская и серверная часть аналитической системы мониторинга Telegram-каналов. Все требования к разработке приложения соблюдены.

					ПГУ 09.03.04– 8БР192.12 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата	Программная реализация аналитической системы мониторинга Telegram-каналов. Пояснительная записка	Лит.	Лист	Листов
Разраб.		Мартынов И.С.					4	75
Пров.		Афонин А.Ю						
Н. Контр.		Такташкин Д.В				Группа 19ВП2		
Утв.		Козлов А.Ю						

Содержание

Введение.....	7
1 Анализ сервисов аналитики Telegram-каналов и выделение требований	9
1.1 Анализ предметной области	9
1.2 Анализ аналитических систем аналогичного назначения	9
1.2.1 Каталог Telegram-каналов и чатов TGStat.....	9
1.2.2 Сервис аналитики Telegram-каналов Telega.in	10
1.2.3 Сервис аналитики Telegram-каналов Telemetr.....	11
1.3 Сравнительный анализ и критериальная оценка аналогов.....	13
1.4 Исследование вариантов использования	14
1.5 Анализ функциональных требований аналитической системы мониторинга Telegram-каналов	17
1.6 Анализ нефункциональных требований аналитической системы мониторинга Telegram-каналов	18
2 Проектирование аналитической системы мониторинга Telegram-каналов.....	20
2.1 Архитектура приложения.....	20
2.2 Проектирование аналитической системы	23
2.3 Выбор технологий и инструментальных средств разработки.....	24
2.4 Проектирование интерфейса.....	27
3 Реализация приложения	31
4 Контроль качества программного обеспечения.....	38
4.1 Выбор режима тестирования	38
4.2 Тетирование аналитической системы методом «черного ящика».....	38
4.3 Тестирование серверной части приложения	40

4.4	Тестирование клиентской части приложения	41
4.5	Метрики кода.....	41
4.5.1	Метрики размера программы.....	44
4.5.2	Метрики Холстеда	44
4.6	Планирование разработки и оценка бюджета.....	46
	Заключение	51
	Список использованных источников	53
	Приложение А. Листинг программы.....	54
	Приложение Б. Результаты тестирования	78

Введение

Telegram является одним из самых популярных кроссплатформенных систем мгновенного обмена сообщениями с функциями обмена текстовыми, голосовыми и видеосообщениями, фотографиями и файлами многих форматов. В Telegram есть массовые группы с тысячами людей, каналы и сообщества. Кроме того, в нем постоянно добавляются новые функции для улучшения пользовательского опыта, автоматизированные боты и многое другое.

На сегодняшний день посредством Telegram распространяется огромное количество информации, наиболее популярными темами являются новости и политика. Зачастую новости и факты, публикуемые в Telegram-каналах, являются непроверенными или заведомо ложными. При отсутствии возможности проверки получаемой информации, необходимо использовать несколько источников и принимать решение, основываясь на результатах анализа.

В связи с этим возникает необходимость в анализе и обобщении информации, предоставляемой различными Telegram-каналами. Этим и определяется значимость и актуальность исследования.

Целью выпускной квалификационной работы является разработка аналитической системы мониторинга Telegram-каналов, которая позволяет оптимизировать получение информации из Telegram.

Объект исследования – средства анализа и обобщения информации, получаемой из Telegram-каналов.

Предмет исследования – программная реализация аналитической системы мониторинга Telegram-каналов.

Для достижения поставленной цели необходимо выполнить следующие задачи:

- провести анализ требований;
- выполнить проектирование программных средств, структур данных и интерфейса;

- реализовать аналитической системы мониторинга Telegram-каналов;
- отладить и протестировать полученное приложение;
- составить план разработки и оценить бюджет разработки.

В работе проведен сравнительный анализ, использованы методы проектирования, разработки, тестирования и метрики кода для аналитической системы.

1 Анализ сервисов аналитики Telegram-каналов и выделение требований

1.1 Анализ предметной области

В настоящее время в Telegram публикуется множество новостей. Аналитическая система Telegram-каналов является отличным инструментом для того, чтобы анализировать новости и публикации.

Благодаря этому приложению пользователи могут получить актуальную информацию о различных событиях, происходящих в интересующей их сфере, а также различные аналитические данные по этой информации.

Исходя из этого, можно сделать вывод, какие функции должны присутствовать в таком приложении:

- просмотр информации о канале;
- просмотр публикаций из разных каналов по категориям;
- просмотр аналитики канала;
- просмотр статистики публикаций.

1.2 Анализ аналитических систем аналогичного назначения

В результате анализа программных продуктов по автоматизации контроля доступа были определены следующие наиболее близкие решения:

- TGStat;
- Telega.in;
- Telemetr.

Эти решения получают высокие оценки от пользователей и пользуются большой популярностью.

Все эти системы предлагают различные функции: детальная аналитика канала, мониторинг количества подписчиков, поиск по ключевым словам.

1.2.1 Каталог Telegram-каналов и чатов TGStat

TGStat – это некоммерческая платформа с открытой статистикой более чем 30 тысяч Telegram-каналов. Сервис обладает широким функционалом и

удобным интерфейсом. Более того, в TGStat можно искать каналы не только по тематикам, но и по ключевым словам, рейтингам и другим параметрам. Кроме того, все каналы попадают в общий список, где также можно добавить фильтры поиска по языковым параметрам, тематике, цене за рекламу и периоду.

Интерфейс сервиса представлен на рисунке 1.

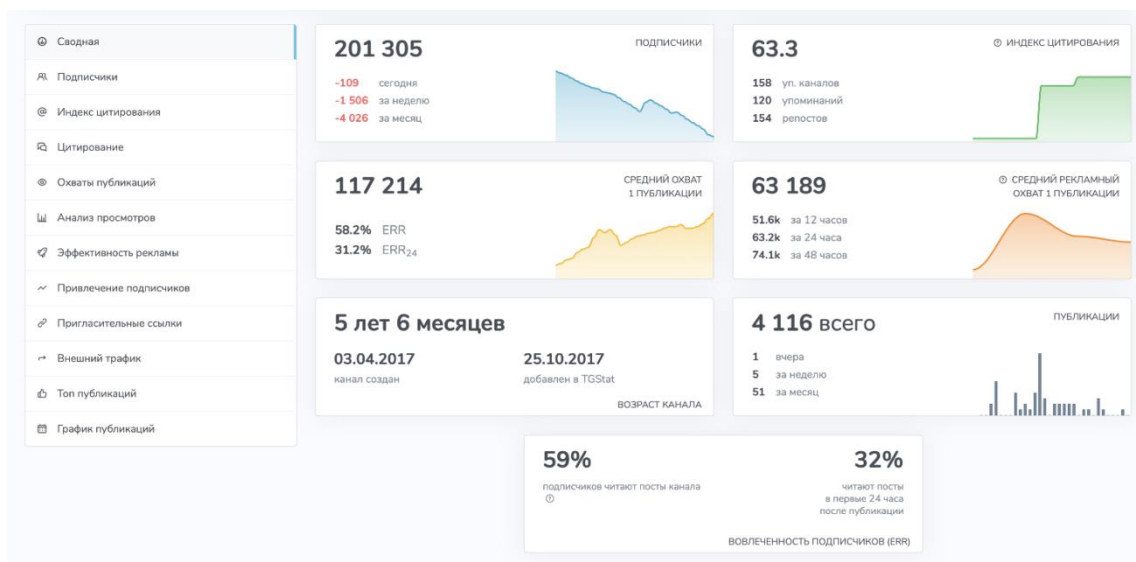


Рисунок 1 – Пример аналитики TGStat

Статистика каждого канала имеет ряд необходимых параметров: количество подписчиков, рост за период, ежедневный охват и средний охват 1 поста, ERR, CCI и лайки. На главной странице канала отображаются самые важные данные. При нажатии на дополнительные вкладки можно увидеть больше аналитики [1].

Основным недостатком приложения являются то, что статистика обновляется не реже, чем раз в 24 часа, что может быть критичным в некоторых ситуациях.

1.2.2 Сервис аналитики Telegram-каналов Telega.in

Telega.in – платформа для поиска каналов для размещения рекламы. Несмотря на то, что в первую очередь сервис выступает посредником между рекламодателем и владельцем площадки, он также обладает широкими возможностями для аналитики Telegram-каналов. Сервис позволяет получать статистические данные и выбирать релевантные платформы, на которых

реклама будет более эффективной. Сюда входят: рейтинг в общем списке платформы, рост подписчиков, вовлеченность аудитории, активность пользователей, история просмотров, график изменения количества подписчиков.

Интерфейс сервиса представлен на рисунке 2.

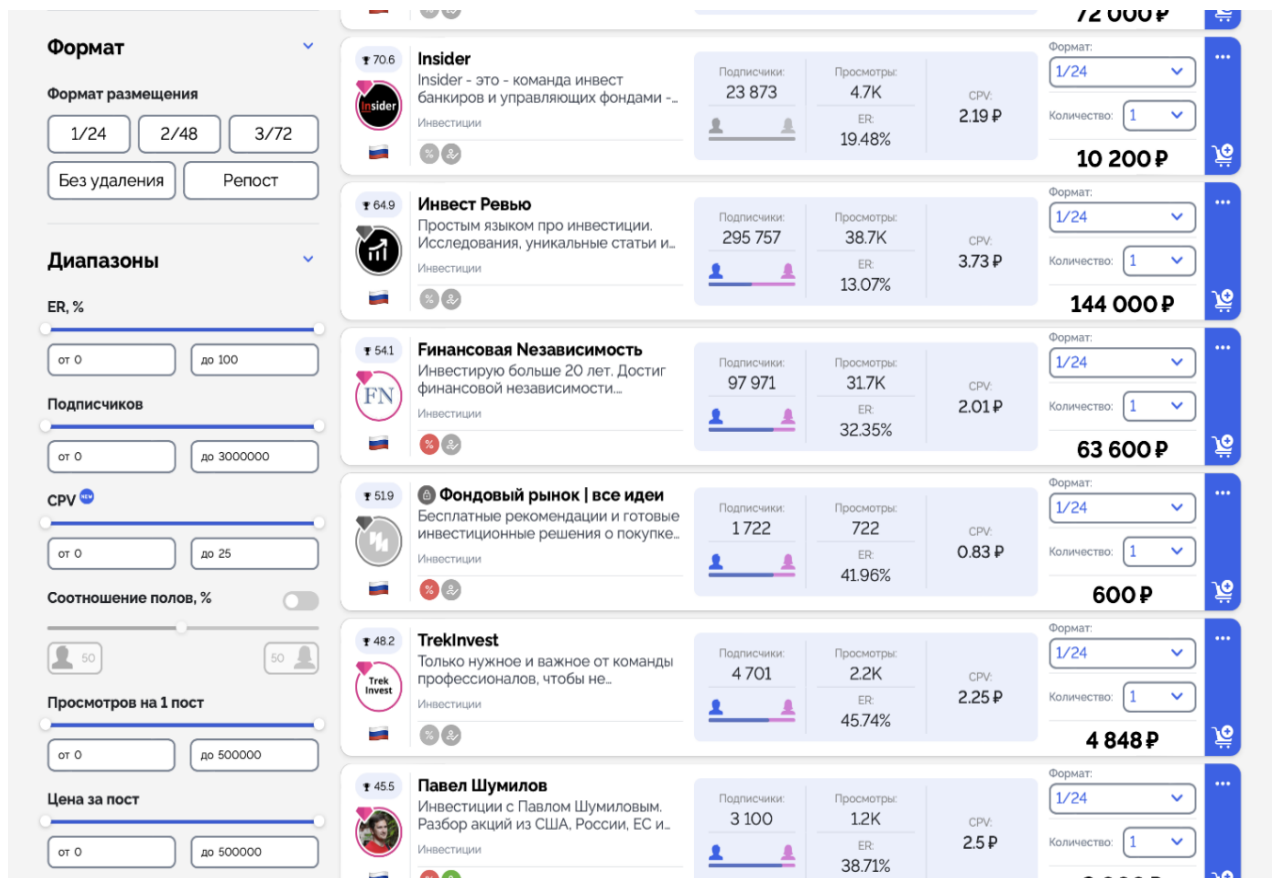


Рисунок 2 – Пример аналитики Telega.in

Недостатком системы является то, что в ней представлены только каналы-партнеры Telega.in.

1.2.3 Сервис аналитики Telegram-каналов Telemetr

Сервис Telemetr был запущен еще в 2017 году, когда Telegram не был так популярен, как сейчас. За это время разработчики платформы реализовали множество полезных функций, среди которых сравнение каналов в рамках единой аналитики, наиболее эффективное время для постинга в конкретном сообществе, поиск контента по записям и так далее.

Интерфейс сервиса представлен на рисунке 3.

Динамика просмотров поста по часам



Просмотры по дням			Просмотры по часам		
День	👁 всего	+ за день	Час	👁 всего	+ за период
2020-04-16	153'716	+153'716	2020-04-16 12:34	Новый пост	
2020-04-17	197'296	+43'580	2020-04-16 12:59	7'317	+7'317
2020-04-18	220'860	+23'564	2020-04-16 13:34	Новый пост	
2020-04-19	235'083	+14'223	2020-04-16 14:34	Новый пост	
2020-04-20	244'409	+9'326	2020-04-16 14:59	70'864	+63'547
2020-04-21	250'420	+6'011	2020-04-16 15:00	Новый пост	
2020-04-22	254'525	+4'105	2020-04-16 15:59	89'697	+18'833
			2020-04-16 16:00	Новый пост	
			2020-04-16 16:57	Новый пост	
			2020-04-16 16:59	103'956	+14'259
			2020-04-16 17:59	113'185	+9'229
			2020-04-16 18:16	Новый пост	
			2020-04-16 18:59	124'232	+11'047
			2020-04-16 20:15	Новый пост	

Рисунок 3 – Аналитика каналов Telemetr

Как и на других платформах, общий поиск в Telemetr позволяет использовать различные фильтры (количество подписчиков, пол аудитории, тип канала и так далее), а также более детально анализировать эти данные при выборе конкретного канала. Это и эффективность рекламы, рост подписчиков после рекламных интеграций, почасовой рост канала (подписки и отписки), размещение контента на других платформах (Youtube, ВКонтакте). Кроме того, здесь же можно использовать функцию, позволяющую отслеживать упоминания бренда, аккаунта, ключевого слова в Telegram.

Одним из главных преимуществ сервиса является его оперативность: данные обновляются каждый час. И уведомления от системы также приходят оперативно. Однако у Telemetr есть ограничения на количество поисковых запросов и результатов поиска.

1.3 Сравнительный анализ и критериальная оценка аналогов

В результате анализа аналогичных программных необходимо подвести итоги. Для проведения сравнительного анализа в качестве критериев, выбраны наиболее важные возможности, которые напрямую влияют на функциональность сервиса, а именно:

- а) функциональные возможности:
 - 1) возможность просмотра детальной аналитики канала;
 - 2) возможность поиска по ключевым словам;
 - 3) возможность просмотра по категориям;
 - 4) возможность анализа ресурсов, на которые ссылается канал;
 - 5) возможность управления списком анализируемых каналов;
- б) нефункциональные возможности:
 - 1) качество интерфейса;
 - 2) бесплатность продукта;
- в) технические возможности:
 - 1) обновление данных в реальном времени;
 - 2) визуализация данных в виде таблицы.

К критериям была выведена ранговая шкала оценки, где используются 0, 1 и 2, чтобы выразить степень реализации функции. Чем выше число, тем выше уровень реализации функции. Например, если функция не реализована вообще, она будет оценена как 0, если реализована частично, то 1, и если полностью реализована, то 2.

Заполним таблицу 1, оценив реализованный функционал описанных выше систем. Затем произведем расчет аддитивной суммы интегральной оценки для каждого сравниваемого продукта по следующей формуле:

$$O_{пп} = \sum_{i=1}^n (Z_i),$$

где $O_{пп}$ – интегральная оценка программного продукта;

n – количество критериев сравнения заявленных продуктов согласно таблице критериев;

Z_i – значение выполнения критерия.

Таблица 1 – Критериальная оценка программных продуктов

№	Критерий	Название программного продукта		
		«TGStat»	«Telega.in»	«Telemetr»
1	возможность просмотра детальной аналитики канала;	2	2	2
2	возможность поиска по ключевым словам;	0	0	2
3	возможность просмотра по категориям;	2	2	1
4	возможность анализа ресурсов, на которые ссылается канал;	0	0	2
5	возможность управления списком анализируемых каналов;	1	1	1
6	качество интерфейса;	1	1	1
7	бесплатность продукта;	2	0	1
8	обновление данных в реальном времени	0	0	1
9	визуализация данных в виде таблицы	2	2	2
	Интегральная оценка	10	8	13

Из данной таблицы можно сделать вывод, что лидером является приложение «Telemetr». Конечно оно, как и все остальные рассматриваемые варианты имеют недостатки в функционале. «TGStat» и «Telega.in» занимают второе и третье место.

1.4 Исследование вариантов использования

Варианты использования определяют, как система должна взаимодействовать с пользователями и что система должна делать для достижения определенных целей [2].

На основе анализа аналогов была сформирована диаграмма вариантов использования, представленная на рисунке 4.

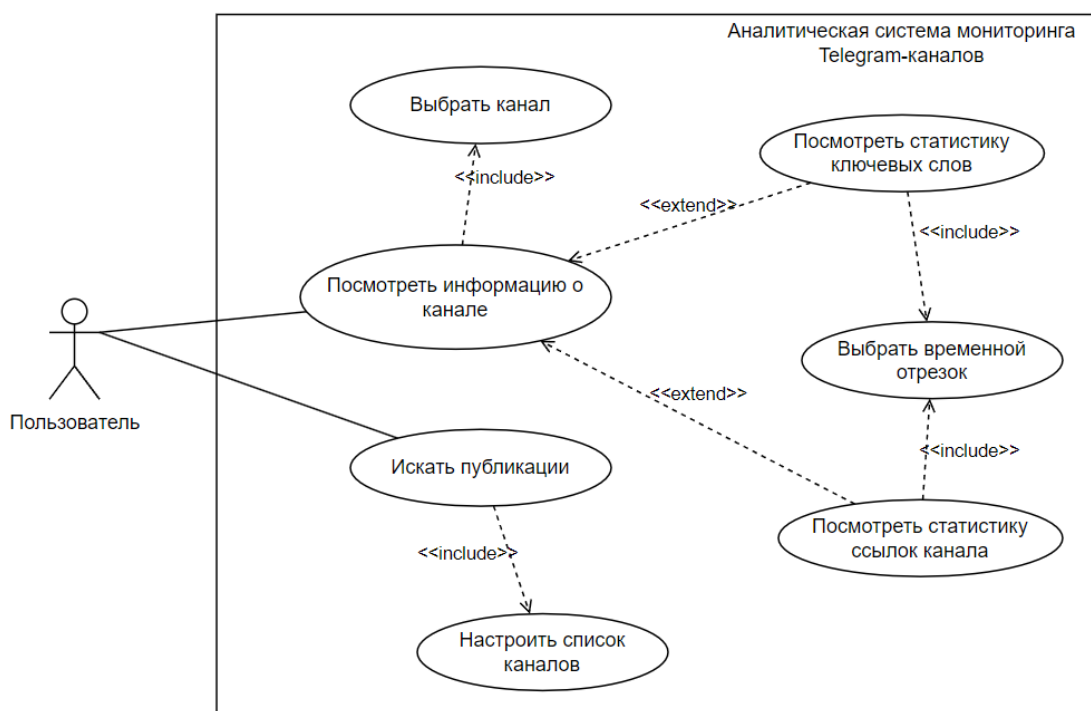


Рисунок 4 – Диаграмма вариантов использования для системы автоматического контроля доступа

На рисунке 5 приведено описание основного порядка выполнения ролью «Пользователь» варианта использования «Посмотреть аналитику канала», позволяющего посмотреть аналитическую сводку выбранного канала.

Название: Посмотреть информацию о канале
ID: 1
Краткое описание: Система предоставляет пользователю возможность посмотреть информацию о telegram-канале.
Главные актеры: Пользователь.
Предусловие: Система загружена.
Основной поток: Система предоставляет пользователю форму для ввода данных. Пользователь вводит ссылку на telegram-канал. Пользователь жмёт кнопку «Поиск». Система проверяет правильность введенной ссылки. Система отображает форму с информацией выбранного канала. Система отображает форму для просмотра статистики ключевых слов. Система отображает форму для просмотра статистики ссылок канала.
Постусловие: Пользователь просмотрел информацию о канале. Система отобразила формы для просмотра статистики.

Рисунок 5 – Спецификация для варианта использования «Посмотреть информацию о канале»

На рисунке 6 приведено описание основного порядка выполнения ролью «Пользователь» варианта использования «Искать публикации», позволяющего найти публикации по ключевым словам.

Название: Искать публикации
ID: 2
Краткое описание: Система предоставляет пользователю возможность найти публикации в каналах по ключевым словам.
Главные актеры: Пользователь.
Предусловие: Система загружена.
Основной поток: Система предоставляет пользователю форму для заполнения списка каналов. Пользователь вводит ссылки на telegram-каналы. Система предоставляет пользователю форму для поиска. Пользователь вводит ключевые слова. Пользователь жмёт кнопку «Поиск». Система проверяет правильность введенных ссылок. Система отображает список публикаций с заданными ключевыми словами.
Постусловие: Пользователь просмотрел список публикаций.

Рисунок 6 – Спецификация для варианта использования «Искать публикации»

На рисунке 7 приведено описание основного порядка выполнения ролью «Пользователь» варианта использования «Просмотреть статистику ключевых слов», позволяющего просмотреть информацию об упоминании заданных ключевых слов.

Название: Просмотреть статистику ключевых слов
ID: 3
Краткое описание: Система предоставляет пользователю возможность найти публикации в каналах по ключевым словам.
Главные актеры: Пользователь.
Предусловие: Система отобразила формы для просмотра статистики.
Основной поток: Пользователь вводит отрезок времени. Пользователь вводит список ключевых слов. Пользователь жмёт кнопку «ОК». Система проверяет правильность введенных данных. Система отображает количество упоминаний каждого ключевого слова.
Постусловие: Пользователь просмотрел статистику ключевых слов.

Рисунок 7 – Спецификация для варианта использования «Просмотреть статистику ключевых слов»

Исследование вариантов использования помогает лучше понять, как должна функционировать система, и позволяет формализовать требования к функциональности системы.

1.5 Анализ функциональных требований аналитической системы мониторинга Telegram-каналов

Разрабатываемое приложение должно получать информацию о каналах с помощью Telegram API. По запросу пользователя должны выводиться запрошенные публикации из выбранных Telegram-каналов. Необходимо производить аналитику отдельно взятого канала: статистика ссылок, которые канал использует в качестве источника, упоминания заданных ключевых слов. Для упорядочения информации должен быть реализован выбор временного отрезка, на котором проводится анализ.

Для разрабатываемого приложения выделены следующие основные требования:

- просмотр детальной аналитики канала – пользователи могут просмотреть аналитику отдельно взятого канала;
- поиск публикаций – пользователи могут искать публикации в каналах по интересующим ключевым словам;
- управление списком каналов – пользователи могут выбирать для просмотра каналы интересующей их категории;
- анализ ссылок – пользователи могут просмотреть, на какие источники и как часто ссылается канал;
- выбор временного промежутка – пользователи могут устанавливать отрезок времени, на котором осуществляется поиск;
- анализ ключевых слов – пользователи увидят, как часто в канале упоминаются заданные ключевые слова.

Приложение будет разработано таким образом, чтобы удовлетворять всем приведенным выше требованиям. Оно будет обеспечивать удобство использования для конечных пользователей, высокую скорость обработки информации. Важно подчеркнуть, что успешная реализация этих требований

потребуется комплексного подхода к разработке, включая тщательное планирование, проектирование и тестирование системы. Это поможет гарантировать, что конечный продукт будет не только функциональным и эффективным, но и безопасным и надежным для использования в реальных условиях.

1.6 Анализ нефункциональных требований аналитической системы мониторинга Telegram-каналов

Система должна обеспечивать быструю обработку запросов к API и взаимодействие с сервером. Это требует эффективного использования ресурсов и оптимизации алгоритмов.

Поскольку разрабатываемое приложение является web-приложением, оно будет разделено на две части:

- серверная часть, обрабатывающая запросы и выполняющая коммуникацию с базой данных;
- клиентская часть (web-браузер), отображающая пользовательский интерфейс и взаимодействующая с серверной частью посредством HTTP-запросов.

Сервер должен быть реализован с помощью PHPStorm, в разработке будет использован язык программирования PHP.

Для реализации статических элементов страниц и шаблонов необходимо использовать язык разметки HTML5 и CSS3.

Часть динамических сценариев на страницах будет реализована при помощи JavaScript. Для изменения стилей страниц будут использованы каскадные таблицы стилей (CSS).

Пользовательский интерфейс должен корректно отображаться при использовании следующих браузеров:

- Google Chrome 60 и выше;
- Firefox 60 и выше;
- Microsoft Edge 88 и выше;
- Safari 12 и выше.

Для выполнения данного требования будет использован CSS-фреймворк Bootstrap 5, позволяющий быстро разработать адаптивный пользовательский интерфейс, поддерживающий большинство современных браузеров.

Для корректной работы приложения аналитической системы мониторинга Telegram-каналов требуется устройство с доступом в интернет. Отдельных требований к мощности системы не предоставляется.

2 Проектирование аналитической системы мониторинга Telegram-каналов

2.1 Архитектура приложения

Для данного приложения было принято решение использовать архитектуру «клиент-сервер». Схема представлена на рисунке 8.

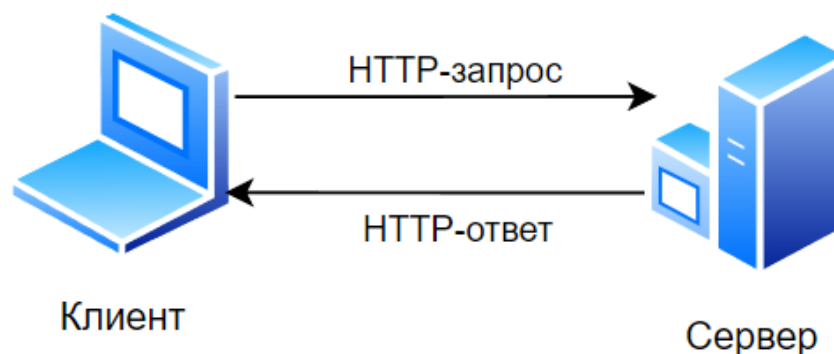


Рисунок 8 – Архитектура «клиент-сервер»

Архитектура клиент-сервер является распространенной моделью для разработки и взаимодействия между приложениями. В этой архитектуре выделяют 2 основных компонента: клиент, сервер.

Клиент-сервер – вычислительная или сетевая архитектура, в которой задания или сетевая нагрузка распределены между поставщиками услуг, называемыми серверами, и заказчиками услуг, называемыми клиентами. Фактически клиент и сервер — это программное обеспечение. Обычно программы расположены на разных вычислительных машинах и взаимодействуют между собой через вычислительную сеть посредством сетевых протоколов, но они могут быть расположены также и на одной машине. Программы-серверы ожидают от клиентских программ запросы и предоставляют им свои ресурсы в виде данных (например, передача файлов посредством HTTP, FTP, потоковое мультимедиа или работа с базами данных) или в виде сервисных функций (например, работа с электронной почтой, общение посредством систем мгновенного обмена сообщениями или просмотр web-страниц во всемирной паутине). Поскольку одна программа-сервер может выполнять запросы от множества программ-клиентов, её размещают на

специально выделенной вычислительной машине, настроенной особым образом, как правило, совместно с другими программами-серверами, поэтому производительность этой машины должна быть высокой. Из-за особой роли такой машины в сети, специфики её оборудования и программного обеспечения, её также называют сервером, а машины, выполняющие клиентские программы, соответственно, клиентами.

HTTP – широко распространённый протокол передачи данных, изначально предназначенный для передачи гипертекстовых документов (то есть документов, которые могут содержать ссылки, позволяющие организовать переход к другим документам).

Протокол HTTP предполагает использование клиент-серверной структуры передачи данных. Клиентское приложение формирует запрос и отправляет его на сервер, после чего серверное программное обеспечение обрабатывает данный запрос, формирует ответ и передаёт его обратно клиенту. После этого клиентское приложение может продолжить отправлять другие запросы, которые будут обработаны аналогичным образом.

Для того, чтобы сформировать HTTP-запрос, необходимо составить стартовую строку, а также задать по крайней мере один заголовок — это заголовок Host, который является обязательным, и должен присутствовать в каждом запросе. Дело в том, что преобразование доменного имени в IP-адрес осуществляется на стороне клиента, и, соответственно, когда вы открываете TCP-соединение, то удалённый сервер не обладает никакой информацией о том, какой именно адрес использовался для соединения.

FTP (File Transfer Protocol) – протокол передачи файлов по сети. Является одним из базовых протоколов Ethernet. Появился в 1971 году и изначально работал в сетях DARPA. На данный момент, как и HTTP, файловая передача основывается на модели, состоящей из совокупности протоколов TCP/IP. Определён в RFC 959.

Основная особенность FTP в том, что он использует двойное подключение. Одно из них используется для передачи команд серверу и происходит по умолчанию через TCP-порт 21, который может быть изменён.

Управляющее соединение существует все время, пока клиент общается с сервером. Канал управления должен быть открыт при передаче данных между машинами. В случае его закрытия передача данных прекращается. Через второе происходит непосредственная передача данных. Оно открывается каждый раз, когда осуществляется передача файла между клиентом и сервером. В случае, если одновременно передаётся несколько файлов, для каждого из них открывается свой канал передачи.

FTP может работать в активном или пассивном режиме, от выбора которого зависит способ установки соединения. В активном режиме клиент создаёт управляющее TCP-соединение с сервером и отправляет серверу свой IP-адрес и произвольный номер клиентского порта, после чего ждёт, пока сервер запустит TCP-соединение с этим адресом и номером порта. В случае, если клиент находится за брандмауэром и не может принять входящее TCP-соединение, может быть использован пассивный режим. В этом режиме клиент использует поток управления, чтобы послать серверу команду PASV, и затем получает от сервера его IP-адрес и номер порта, которые затем используются клиентом для открытия потока данных со своего произвольного порта.

При реализации приложения будет использован протокол HTTP.

Клиент – это приложение или устройство, которое инициирует запросы к серверу для получения данных или выполнения определенных действий. Клиент может быть веб-браузером, мобильным приложением, настольным приложением и т. д.

Сервер – это центральный компонент архитектуры, который обрабатывает запросы от клиентов и предоставляет им необходимую функциональность или данные. Сервер может быть физическим сервером или набором серверов, которые обрабатывают запросы параллельно.

Взаимодействие между клиентом и сервером осуществляется через HTTP-запросы, при этом сервер осуществляет запросы к Telegram API.

API (Application Programming Interface) – определяет набор правил и протоколов, по которым клиент и сервер взаимодействуют друг с другом. API предоставляет клиенту определенные методы и функции, которые могут быть

вызваны для выполнения определенных операций или доступа к данным на сервере.

В данной работе клиент и сервер взаимодействуют посредством вызовов API. Клиент отправляет запросы к серверу, используя определенные API-методы, а сервер обрабатывает эти запросы и возвращает соответствующие ответы.

Преимущества такой архитектуры включают модульность, масштабируемость и возможность распределения ответственности между клиентом и сервером. Клиент и сервер могут разрабатываться и масштабироваться независимо друг от друга, что облегчает разделение функциональности и обновление системы. API также позволяет разным приложениям и сервисам интегрироваться и обмениваться данными между собой, обеспечивая гибкость и расширяемость системы [3].

2.2 Проектирование аналитической системы

После проведения анализа предметной области и выработки требований была построена диаграмма деятельности для варианта использования «Искать публикации». Данная диаграмма представлена на рисунке 9.

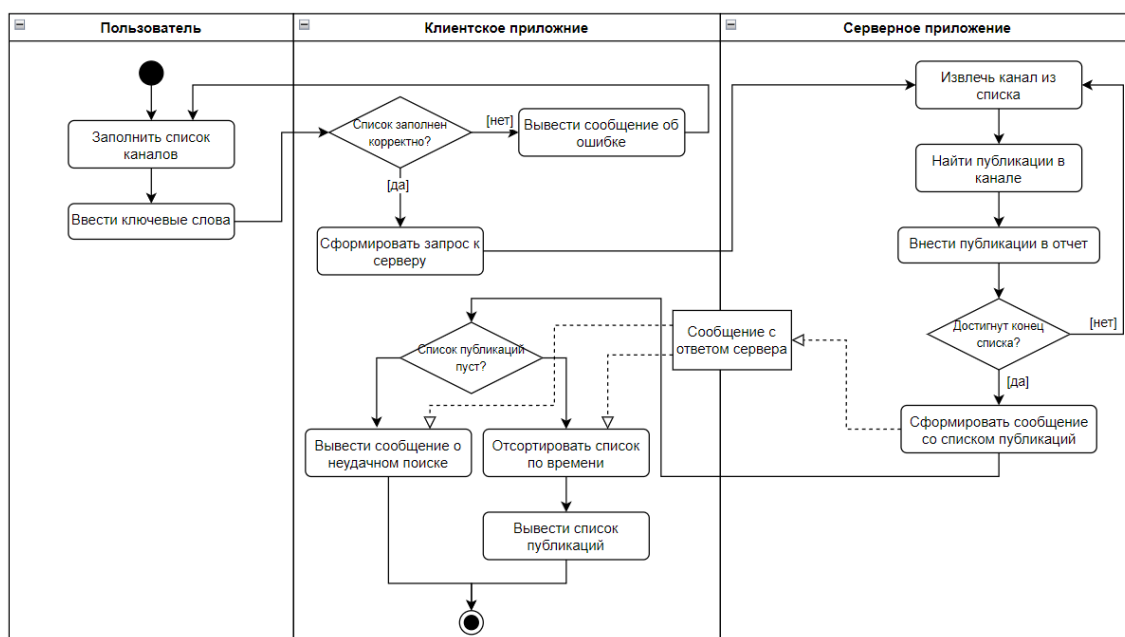


Рисунок 9 – Диаграмма деятельности для варианта использования «Искать публикации»

Диаграмма деятельности в нотации UML предоставляет возможность представить поведение системы в виде последовательности действий, позволяя легко визуализировать и анализировать процессы и взаимодействие между объектами в системе.

Запуск процесса происходит, когда пользователь запускает приложение и открывает вкладку «Публикации». После заполнения списка ссылок на каналы и ввода ключевых слов для поиска, клиентское приложение проверяет правильность ввода, формируя соответствующее сообщение в случае ошибки. Затем формируется запрос к серверу, содержащий входные данные. Серверное приложение принимает запрос и извлекает ссылку на канал, находящуюся во главе списка. После этого происходит поиск публикаций в канале с соответствующими ключевыми словами путем запросов к Telegram API. Найденные публикации добавляются в отчет. Если не был достигнут конец списка, извлечение ссылки на канал происходит снова. Таким образом, циклически производится поиск во всех каналах из списка. После этого серверное приложение формирует сообщение, содержащее все найденные публикации. В случае если публикаций найдено не было, формируется сообщение о неудачном поиске. Если же в отчете есть публикации, клиентское приложение сортирует их по дате и времени и выводит на соответствующую форму. После этого процесс завершается.

2.3 Выбор технологий и инструментальных средств разработки

В процессе анализа предметной области было решено использовать Rational Unified Process (RUP) методологию. RUP – это гибкая методология разработки программного обеспечения, разработанная компанией Rational Software. RUP представляет собой итеративную и инкрементальную методологию, ориентированную на объектно–ориентированную разработку. Использование методологии RUP направлено на итеративную модель разработки. Особенность методологии состоит в том, что степень формализации может меняться в зависимости от потребностей проекта. Можно по окончании каждого этапа и каждой итерации создавать все требуемые

документы и достигнуть максимального уровня формализации, а можно создавать только необходимые для работы документы, вплоть до полного их отсутствия. За счет такого подхода к формализации процессов методология является достаточно гибкой и широко популярной. Данная методология применима как в небольших и быстрых проектах, где за счет отсутствия формализации требуется сократить время выполнения проекта и расходы, так и в больших и сложных проектах, где требуется высокий уровень формализма, например, с целью дальнейшей сертификации продукта. Это преимущество дает возможность использовать одну и ту же команду разработчиков для реализации различных по объему и требованиям.

RUP использует итеративную модель разработки. В конце каждой итерации (в идеале продолжающейся от 2 до 6 недель) проектная команда должна достичь запланированных на данную итерацию целей, создать или доработать проектные артефакты и получить промежуточную, но функциональную версию конечного продукта. Итеративная разработка позволяет быстро реагировать на меняющиеся требования, обнаруживать и устранять риски на ранних стадиях проекта, а также эффективно контролировать качество создаваемого продукта.

Применение RUP позволяет командам разработчиков эффективно управлять проектами, управлять рисками и обеспечивать качество продукта [4].

Определяет следующие четыре основные фазы разработки:

- а) начальная фаза (inception phase):
 - 1) определение целей проекта и его ограничений;
 - 2) анализ бизнес-требований и определение основных функциональных и нефункциональных требований;
 - 3) определение архитектуры системы;
 - 4) оценка рисков и планирование проекта;
 - 5) создание предварительного плана разработки и оценка затрат;
- б) фаза разработки (elaboration phase):
 - 1) уточнение требований и архитектуры системы;
 - 2) разработка более детального плана разработки;

- 3) создание прототипов и проведение исследований концепции;
- 4) идентификация и управление рисками;
- 5) определение основных компонентов и модулей системы;
- 6) разработка итерационного плана разработки;
- в) фаза конструирования (construction phase):
 - 1) итеративная реализация системы, включая программирование, тестирование и интеграцию компонентов;
 - 2) систематическое выполнение итераций разработки, в каждой из которых происходит детализация требований, проектирование, разработка и тестирование;
 - 3) постепенное улучшение системы и добавление новых функций;
 - 4) контроль качества и тестирование продукта;
- г) фаза перехода (transition phase):
 - 1) подготовка системы к выпуску и передаче заказчику;
 - 2) тестирование системы на соответствие требованиям и стабильность работы;
 - 3) обучение пользователей и администраторов системы;
 - 4) поддержка системы после выпуска, включая исправление ошибок и обработку запросов на изменение;
 - 5) оценка проекта и извлечение уроков для будущих проектов.

RUP предлагает разработку через серию итераций, что позволяет команде получать обратную связь от заказчика и пользователей на ранних стадиях проекта. Это позволяет быстро внести корректировки и уточнения в требования и архитектуру, улучшая конечный результат. Также выбранная методология предоставляет гибкую рамку для разработки, позволяя адаптировать процесс к конкретным потребностям проекта. Это позволяет выбирать и масштабировать фазы, активности и роли в соответствии с требованиями и ограничениями. RUP акцентирует внимание на идентификации и управлении рисками в проекте.

Предлагаются подходы к анализу и оценке рисков. Это позволяет сосредоточиться на критических аспектах проекта и своевременно реагировать

на возникающие проблемы. Методология обладает механизмами для управления изменениями в проекте. Это помогает контролировать и управлять изменениями в требованиях, архитектуре и планах проекта, чтобы минимизировать негативные последствия и обеспечить согласованность и качество продукта. Акцентируется особое внимание на качество разрабатываемого программного обеспечения. Также RUP акцентирует внимание на контроле качества во время всего процесса разработки.

При планировании проекта было решено использовать Microsoft Project для эффективного управления всеми стадиями его выполнения. Это приложение позволяет структурировать проект, составить его календарный план и рассчитать затраты труда, материальных и финансовых ресурсов в процессе работы над проектом.

2.4 Проектирование интерфейса

Проектирование интерфейса – это задача, основанная на разработке визуального представления и структуры интерактивных элементов продукта (например, веб-сайта, мобильного приложения, программного обеспечения), с целью обеспечения максимального удобства и эффективности взаимодействия пользователя с продуктом.

Пользовательский интерфейс, или UI (User Interface) — это внешний вид продукта, способ общения между пользователем и программой. А ещё интерфейс влияет на то, будет ли продукт приносить деньги и пользоваться уважением и любовью аудитории.

Проектирование интерфейса включает в себя понимание потребностей, поведения и ожиданий целевой аудитории, определение структуры, расположения элементов управления и навигационных схем для обеспечения удобства использования. Затем прототипы тестируются на пользователях, чтобы определить, насколько удобен интерфейс для использования. На основании обратной связи интерфейс может быть доработан и улучшен.

Интерфейс приложения должен быть интуитивно понятным и привлекательным для пользователя. Интерфейс должен реализовывать

выделенные требования в полной мере. Интерфейс должен быть организован так, чтобы облегчить пользователю взаимодействие с приложением. Это означает, что элементы управления должны быть легко доступны, их функции должны быть понятны из контекста, а ошибка ввода пользователя должна вызывать понятные и информативные сообщения об ошибках. Интуитивность интерфейса означает, что пользователь может понять, как использовать приложение, без необходимости обращения к руководствам или справке. Дизайн интерфейса должен быть приятным и профессиональным, с хорошим сочетанием цветов, шрифтов и изображений.

Все функции и возможности, которые были определены в процессе анализа требований, должны быть реализованы в интерфейсе приложения. Пользователь должен иметь возможность выполнить все задачи, для которых приложение было разработано, через доступные в интерфейсе инструменты и функции.

Для правильной реализации интерфейса необходимо составить макеты. Макет вкладки «Публикации» представлен на рисунке 10.

Публикации		Каналы	
Каналы		Публикации	
link	Удалить	Ключевые слова	Поиск
Ссылка на канал	Добавить	Name: date time	
		Текст публикации	

Рисунок 10 – Макет интерфейса вкладки «Публикации»

Панель вкладок позволяет быстро переключаться между вкладками «Публикации» и «Каналы». Контейнер «Каналы» представляет собой панель, позволяющую редактировать список каналов, в которых производится поиск. Внутри него располагаются поля для ввода ссылок на Telegram-каналы так, что в любой момент можно добавить новую ссылку или удалить уже добавленную. Контейнер «Публикации» содержит поле ввода ключевых слов для поиска, а также сами публикации, которые являются результатами поиска. В заголовке публикаций указывается название канала, дата и время публикации.

Макет вкладки «Каналы» представлен на рисунке 11.

The mockup shows the 'Каналы' (Channels) tab selected. At the top, there's a dark navigation bar with two tabs: 'Публикации' and 'Каналы'. Below this is a search bar with a text input containing 'link' and a 'Поиск' (Search) button. The main content area is divided into three sections:

- Описание канала (Channel Description):** A section with a header 'Описание канала' and a large text input field labeled 'Текст описания'.
- Ссылки (Links):** A section with a header 'Ссылки'. It contains a date range selector: 'Период: с [date] [calendar icon] по [date] [calendar icon] [OK]'. Below this is a table titled 'Упоминания ссылок:' with columns 'link' and 'count'.
- Ключевые слова (Keywords):** A section with a header 'Ключевые слова'. It contains a date range selector: 'Период: с [date] [calendar icon] по [date] [calendar icon] [OK]'. Below this is a table titled 'Упоминания ключевых слов:' with columns 'Keyword' and 'count'. There are also two input fields at the bottom: 'Keyword' with a 'Удалить' (Delete) button, and 'Ключевое слово' with a 'Добавить' (Add) button.

Рисунок 11 – Макет интерфейса вкладки «Каналы»

Наверху располагается поле для ввода ссылки на исследуемый Telegram-канал. Под ним выводится имя канала и его количество подписчиков. Контейнер «Описание канала» содержит стандартное описание канала Telegram. Контейнер «Ссылки» Представляет собой форму для отправки

запроса на статистику используемых в канале ссылок. Внутри этого контейнера располагаются поля для ввода исследуемого отрезка времени. Результат запроса выводится в текстовом формате как пара ссылка: количество. Контейнер «Ключевые слова» представляет собой форму для отправки запроса на статистику упоминаний в канале ключевых слов. Внутри него находятся поля для ввода исследуемого отрезка времени, а также поля для ввода ключевых слов для поиска. Результат выводится в текстовом формате как пара ключевое слово: количество.

Таким образом, был спроектирован простой и удобный пользовательский интерфейс, позволяющий быстро взаимодействовать с системой.

3 Реализация приложения

На текущий момент существует два основных инструмента API, с помощью которых можно задействовать сервисы Telegram: Telegram Bot API, предназначенный для создания чат-ботов, и Telegram API, позволяющий создавать собственный Telegram клиенты. При реализации приложения будет использован Telegram API, поскольку с его помощью можно извлекать информацию из каналов посредством запросов, используя собственный клиент Telegram.

Telegram API — это набор программных интерфейсов, которые позволяют разработчикам взаимодействовать с серверами Telegram и создавать приложения, которые могут отправлять и получать сообщения, изображения и файлы. С помощью Telegram API разработчики могут создавать ботов, которые могут автоматизировать различные задачи, включая поддержку клиентов, доставку новостей и многое другое. Кроме того, Telegram API позволяет разработчикам создавать интеграции со сторонними сервисами, такими как чат-боты, социальные сети и платформы электронной почты.

Telegram API построен на клиент-серверной архитектуре. Клиенты отправляют запросы на серверы Telegram, которые обрабатывают запросы и возвращают ответы. Запросы и ответы передаются через Интернет с использованием HTTP или MTProto.

Чтобы использовать API Telegram, разработчики должны сначала создать бота Telegram. Бот — это приложение, которое может получать сообщения, отправленные пользователями, и отвечать на них. Чтобы создать бота, разработчики должны сначала зарегистрировать новую учетную запись бота в BotFather, боте Telegram, который может помочь создавать ботов и управлять ими. После регистрации бота разработчики могут получить токен API, который они могут использовать для аутентификации своих запросов к Telegram API.

После создания бота разработчики могут использовать Telegram API для отправки и получения сообщений, изображений и файлов. Сообщения можно отправлять отдельным пользователям, группам или каналам. Разработчики также могут использовать Telegram API для управления настройками бота, включая изображения профилей, описания и команды.

Администраторам групп и чатов порой необходимо пользоваться таким полезным инструментом как боты, а кому-то и вовсе требуется свой клиент для запросов серверу Telegram. Для взаимодействия с ботами существует немалое количество готовых библиотек для различных языков и платформ, но если требуется написать собственную программу или библиотеку (возможно, написанные другими людьми библиотеки ограничены и не содержат требуемых вам функций), то для этого Telegram имеет собственный API.

Telegram поддерживает два варианта API: Telegram Bot API (обычный бот) и Telegram API (собственный клиент или юзербот). В сегодняшнем материале речь пойдет о втором случае, когда нужно свой аккаунт использовать как бота или же для каких-либо иных действий. Например, с помощью Telegram API можно написать полностью свой клиент, который будет работать точно также как и основной "телеграммовский" клиент, можно реализовать отображение реального времени на вашей аватарке, можно использовать своего юзербота как сканер для защиты вашего чата от спама, флуда, "рейдов" и т. д. В общем, применений найдется много.

Протокол Telegram известен своей доступностью и открытостью. У него есть множество публичных реализаций: `tdlib/td`, `rubenlagus/TelegramApi`, `vysheng/tg`, `LonamiWebs/Telethon` и другие. Имея в распоряжении такой богатый инструментарий, можно решать многие прикладные задачи, в том числе реализовать всестороннюю аналитику Telegram-каналов [5].

Официальный клиент Telegram содержит в себе массу API-цепочек, которые реализуют определенные пользовательские сценарии. Взаимодействие на основе сценариев – наиболее удобный и предпочтительный способ, поскольку он позволяет избежать использования повторного кода, поэтому

приложение будет реализовано путем упрощения взаимодействия с Telegram на основе использования сценариев.

Сценарий — это «черный ящик», реализованный в виде последовательности API-вызовов, позволяющей достичь определенной конечной цели (output) на основе входных параметров (input). Аргументы представляют собой понятные сущности окружающего мира (например, ключевое слово для поиска).

В результате использования аргументов внутри черного ящика на выходе получается запрошенная информация (например, сообщения с указанным ключевым словом). Все взаимодействие с Telegram API при этом инкапсулировано в реализации сценария. Сценарии могут сочетаться друг с другом, формируя более комплексные сценарии.

Сервер приложения играет центральную роль в архитектуре веб-приложения. Он обрабатывает входящие запросы от клиентов (обычно браузеров или других веб-приложений), выполняет необходимую бизнес-логику и возвращает ответы.

Чтобы использовать Telegram API, разработчики должны сначала получить токен API от BotFather. Токен API — это уникальный идентификатор, который разработчики могут использовать для аутентификации своих запросов к Telegram API. После получения токена API разработчики могут использовать любой язык программирования, поддерживающий HTTP-запросы, для взаимодействия с Telegram API.

API Telegram предоставляет множество способов отправки и получения сообщений, управления настройками бота и т. д. Например, разработчики могут использовать метод `sendMessage` для отправки текстовых сообщений пользователям или группам. Метод `sendPhoto` можно использовать для отправки изображений, а метод `sendDocument` — для отправки файлов.

Помимо отправки и получения сообщений, API Telegram также позволяет разработчикам управлять настройками бота, такими как изображения профиля, описания и команды. Метод `setWebhook` можно использовать для настройки

веб-перехватчика, который позволяет API Telegram отправлять обновления на URL-адрес всякий раз, когда поступает новое сообщение.

API Telegram предоставляет разработчикам инструменты, необходимые им для создания ботов и интеграций, которые могут взаимодействовать с пользователями Telegram. С помощью Telegram API разработчики могут автоматизировать множество задач, включая поддержку клиентов, доставку новостей и многое другое. API доступен как в версии HTTP, так и в версии MTProto, и предоставляет множество способов отправки и получения сообщений, управления настройками бота и т. д. Если вы разработчик, который хочет создать бота или интеграцию Telegram, API Telegram — это важный инструмент, который должен быть в вашем наборе инструментов.

Клиент приложения предоставляет пользователю дружелюбный интерфейс и посылает запросы на сервер.

Предоставляемые сценарии представлены в таблице 2.

Таблица 2 – Спецификация сценариев приложения

№	Имя сценария	Назначение	Входные данные	Выходные данные
1	parseChannelLinks	Подсчет всех ссылок, опубликованных в канале за определенный период.	ссылка на канал, начало временного отрезка, конец временного отрезка.	Список пар значений ссылка – количество.
2	parseChannelMessages	Возврат всех публикаций в канале за определенный период.	ссылка на канал, начало временного отрезка, конец временного отрезка.	Список публикаций.
3	searchChannelMessages	Поиск публикаций, содержащих ключевое слово, в канале за определенный период.	ссылка на канал, начало временного отрезка, конец временного отрезка, ключевое слово.	Список публикаций.
4	countChannelMessages	Подсчет публикаций, содержащих ключевое слово, в	ссылка на канал, начало временного отрезка, конец	Список пар значений ключевое слово – количество.

		канале определенный период.	за временного отрезка, ключевое слово.	
5	getChannelDescription	Возврат описания канала.	ссылка на канал.	описание канала.

Диаграмма компонентов является частью UML (Unified Modeling Language), которая используется для визуализации структуры системы и показывает программные компоненты, связи между ними и их взаимодействия.

В контексте сервера приложения, диаграмма компонентов может использоваться для иллюстрации компонентов, связей, интерфейсов. Это могут быть модули, классы, объекты или другие элементы системы, которые выполняют определенные функции. Например, в приложении могут быть компоненты для управления пользователями, обработки запросов и управления данными. Например, компонент для обработки запросов может взаимодействовать с компонентом управления пользователями для аутентификации запроса.

Диаграмма компонентов сервера приложения полезна, поскольку она позволяет разработчикам и стейкхолдерам понять структуру и взаимодействия системы на высоком уровне. Это может быть полезно при проектировании системы, внесении изменений и устранении неполадок.

На рисунке 12 представлена диаграмма компонентов сервера приложения.

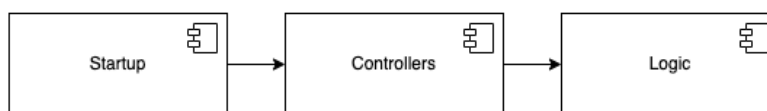


Рисунок 12 – Диаграмма компонентов сервера приложения

Ниже представлено описание каждого компонента:

- компонент Startup – отвечает за логику запуска сервера в сети; благодаря нему приложение может получать доступ по URL;

- компонент **Controllers** – промежуточный компонент, предназначенный для связи с логикой приложения; сюда включены контроллеры;
- компонент **Logic** – реализует логику приложения, а именно доступ к базе данных.

Для корректной работы мобильного приложения в связке с сервером и базой данных, необходимо правильно развернуть все компоненты.

Диаграмма развертывания в UML (Unified Modeling Language) обычно используется для визуализации физической архитектуры и развертывания программного обеспечения [6]. Это один из видов структурных диаграмм в UML, которые показывают, как система будет установлена и функционировать на аппаратном обеспечении. Узел представляет собой физическую машину, на которой выполняется часть программного обеспечения. Это может быть сервер, на котором работает ваше веб-приложение, или база данных, на которой хранятся данные. Артефакты представляют собой физические информационные единицы, которые используются и создаются в процессе разработки программного обеспечения. Это может быть исполняемый файл, скрипт, документация и т. д. Зависимости показывают, как различные узлы и артефакты связаны друг с другом.

Для этого была разработана диаграмма развертывания, представленная на рисунке 13.

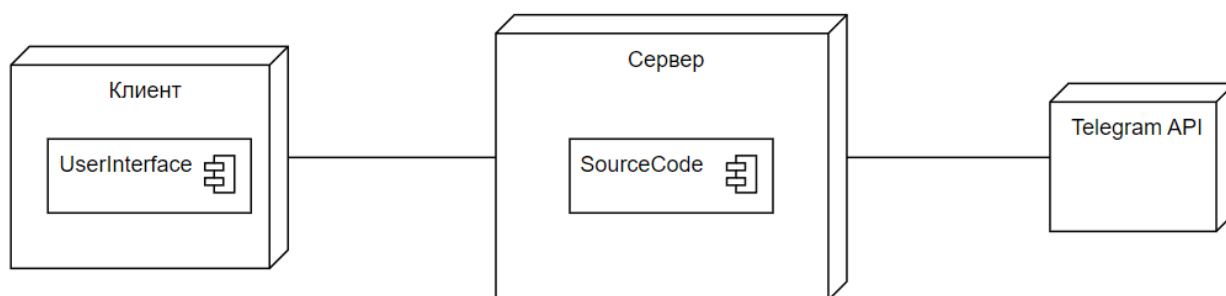


Рисунок 13 – Диаграмма развертывания приложения

Диаграмма развертывания приложения полезна, поскольку она позволяет лучше понять, как и где будет выполняться ваше программное обеспечение. Это особенно полезно при планировании инфраструктуры и развертывания, поскольку она показывает, какие ресурсы будут использоваться, как они будут взаимодействовать друг с другом.

Листинг реализованной аналитической системы представлен в приложении А.

4 Контроль качества программного обеспечения

4.1 Выбор режима тестирования

Тестирование программного обеспечения – это процесс оценки системы или ее компонентов с целью определить, соответствует ли она заданным требованиям. Этот процесс включает в себя запуск программного обеспечения в различных сценариях и условиях для выявления ошибок, проблем с производительностью или несоответствий предварительно определенным стандартам и требованиям [7].

Существуют различные методы и уровни тестирования, включая модульное тестирование, интеграционное тестирование, системное тестирование и приемочное тестирование. Для разработанного приложения было выбрано функциональное тестирование методом черного ящика, так как оно позволяет рассмотреть продукт от лица пользователя. Функциональное тестирование позволяет проверить поведение системы на соответствие ожидаемым результатам.

Функциональное тестирование направлено на проверку соответствия функционала программного продукта заранее определенным требованиям. Данный метод тестирования является одним из самых применяемых, так как позволяет наиболее полно проверить программу на соответствие требованиям.

Цель функционального тестирования – гарантировать, что разработанный продукт функционирует в соответствии с требованиями и ожиданиями.

Было принято решение провести функциональное тестирование, так как оно сосредоточено на проверке того, выполняет ли система свои предполагаемые функции и задачи. Этот вид тестирования важен, так как он напрямую относится к требованиям и ожиданиям пользователей от системы.

4.2 Тестирование аналитической системы методом «черного ящика»

Для разработанного проекта будет проведено тестирование методом черного ящика, так как оно в достаточной мере позволяет протестировать приложение и не требует больших затрат для реализации.

Тестирование методом черного ящика – это метод тестирования программного обеспечения, при котором тестируются входы и выходы программы без знания внутренней структуры и реализации системы. В этом методе тестируется только внешнее поведение программы, а не ее внутренняя логика [8].

Для примера будут приведены тесты для метода по вводу ссылки на канал. Табличное представление классов эквивалентности приведены в таблице 3.

Таблица 3 – Классы эквивалентности

	Правильные КЭ	Неправильные КЭ
Длина ссылки	18-45 (1)	0 – 17 (2) 46 – ∞ (3)
Структура ссылки	Начинается с https://t.me/ (4)	Начинается не с https://t.me/ (5)
Содержание ссылки	Буквы латинские(6) Цифры(7) Подчеркивание(8) Дефис(9) Двоеточие(10) Косая черта(11) Точка(12)	Остальные символы(13)

После определения структуры для ввода ссылки на канал необходимо привести примеры, которые будут содержать уникальную последовательность классов эквивалентности. Данные примеры в полной мере будут покрывать метод по созданию логина. Примеры приведены в таблице 4.

Таблица 4 – Тест – кейсы ввода ссылки на канал

№	Классы эквивалентности	Входные данные	Выходные данные
1	Все правильные 1,4,6,7,10,11,12	https://t.me/cat0news	Успешно
2	Все правильные 1,4,6,8,10,11,12	https://t.me/mopevm_pnzgu	Успешно
3	3,4,6,9,10,11,12	https://t.me/the-link-of-the-channel-is-too-long	Ошибка: больше 46 символов
4	2,4,6,10,11,12	https://t.me/	Ошибка: меньше 17 символов
5	1,5,6,10,11,12	https://vk.com/pnzgu	Ошибка: неверное начало ссылки
6	1,4,6,10,11,12,13	https://t.me/some*channel	Ошибка: символ, не входящий в список разрешенных

Как видно из таблицы примеры полностью покрывают условия для ввода ссылки на канал.

4.3 Тестирование серверной части приложения

Для основных вариантов использования были проведены функциональные тесты. Тестирование серверной части приложения – это процесс, который проверяет работоспособность и надежность функционала, который выполняется на стороне сервера. В случае обнаружения ошибок, их исправляют и проводят повторное тестирование.

Тестирование серверной части приложения необходимо, чтобы гарантировать, что она будет работать корректно и надежно в реальных условиях. Это помогает предотвратить проблемы, которые могут возникнуть у пользователей, и обеспечивает высокое качество продукта.

Серверная часть приложения играет ключевую роль в обеспечении работоспособности системы, поэтому ее тестирование необходимо для обеспечения качества продукта.

Функциональные тесты серверной части приложения фокусируются на проверке корректности выполнения функций, которые предоставляет сервер. Эти тесты убеждаются, что серверное приложение выполняет все требуемые операции и возвращает ожидаемые результаты.

Назначение, описание, ожидаемые и полученные результаты приведены в таблице 5.

Таблица 5 – Функциональные тесты серверной части приложения

№ теста	Назначение	Описание	Входные данные	Ожидаемый результат	Актуальный результат
1	Поиск публикаций	Поиск публикаций по ключевому слову в нескольких заданных Telegram-каналах	Ссылки на Telegram-каналы, ключевое слово	Вывод публикаций на форму	Пройден (рисунок Б.1)
2	Поиск канала по ссылке	Поиск канала по ссылке	Ссылка на Telegram-канал	Вывод описания канала на форму	Пройден (рисунок Б.2)
3	Запрос аналитики ссылок	Запрос аналитики по ссылкам, публикуемум в канале на отрезке времени	Ссылка на Telegram-канал, временной отрезок	Успешный вывод аналитики ссылок	Пройден (рисунок Б.3)
4	Запрос аналитики ключевых слов.	Запрос аналитики по заданным ключевым словам, публикуемум в канале на отрезке времени	Ссылка на Telegram-канал, временной отрезок, ключевые слова	Успешный вывод аналитики ключевых слов	Пройден (рисунок Б.4)

Исходя из результатов, полученных в ходе тестирования, можно сделать вывод, что функциональные требования реализованы в полной мере.

4.4 Тестирование клиентской части приложения

Для основных функций, представленных на диаграмме вариантов использования, были проведены функциональные тесты. Тестирование клиентской части приложения включает в себя проверку пользовательского интерфейса и взаимодействия пользователя с системой. Это может включать проверку того, что страницы загружаются корректно, элементы интерфейса отображаются и функционируют, как ожидается, и данные, введенные пользователем, обрабатываются корректно.

Их перечень и описание представлены в таблице 6.

Таблица 6 – Перечень тестов клиентской части

№ теста	Назначение	Описание	Входные данные	Ожидаемый результат	Актуальный результат
1	Поиск канала	Пользователь пытается найти канал по ссылке	Ссылка на Telegram-канал	На форму выводится название и описание канала, становятся доступны аналитические формы	Пройден (рисунок Б.5)
2	Поиск публикации	Пользователь путается найти публикации в каналах.	Недействительная ссылка на канал.	Вывод сообщения о неудачном поиске.	Пройден (рисунок Б.6)

Исходя из результатов, полученных в ходе тестирования, можно сделать вывод, что функциональные требования реализованы в полной мере, в том числе и обработка исключительных ситуаций.

4.5 Метрики кода

Метрики кода отображают количественные величины программного обеспечения, которые предоставляют разработчику доступ к таким свойствам, как эффективность, оптимизацию и масштабируемость написанного кода приложения [9].

Способы, которые используют для получения метрик код, существует огромное количество. Данные способы различны по своим алгоритмам и по информации, полученной в виде результата. Для разработанного приложения будут проведены следующие группы метрик:

- метрики размера программы (SLOC-метрики);
- метрики сложности потока управления программ (метрики Холстеда).

Для оценки возьмем сценарий `parseChannelLinks`, который используется для подсчета ссылок, опубликованных в канале за определенный период. Код приведен ниже в листинге 1.

Листинг 1 – Подсчет ссылок, опубликованных в канале за определенный период

```
<?php

/** @noinspection BypassedUrlValidationInspection */

declare(strict_types=1);

require_once __DIR__.'../../vendor/autoload.php';

$argsOrFalse = getopt('g:l:f:t:h', ['group-id:', 'deep-link:',
'timestamp-from:', 'timestamp-to:', 'help']); //аргументы
$hasGroupId = array_key_exists('g', $argsOrFalse) ||
array_key_exists('group-id', $argsOrFalse);
$hasLink = array_key_exists('l', $argsOrFalse) ||
array_key_exists('deep-link', $argsOrFalse);

if ($hasGroupId && $hasLink) {
    echo 'Group ID and Deep Link are mutually exclusive. Please
specify only one of them.'.PHP_EOL;

    exit(1);
}

$groupId = $argsOrFalse['g'] ?? $argsOrFalse['group-id'] ?? null;
$deepLink = $argsOrFalse['l'] ?? $argsOrFalse['deep-link'] ?? '';
$timestampStart = $argsOrFalse['f'] ?? $argsOrFalse['timestamp-
from'] ?? null;
$timestampEnd = $argsOrFalse['t'] ?? $argsOrFalse['timestamp-to']
?? null;

$timestampStart = $timestampStart === null ? null : (int)
$timestampStart;
```

```

$timestampEnd = $timestampEnd === null ? null : (int)
$timestampEnd;

$generator = new ReusableClientGenerator();
$request = $groupId !== null
    ? GroupRequest::ofGroupId((int) $groupId)
    : GroupRequest::ofUserName($deepLink);

$result = [];
$parsedLinks = static function (
    /** @noinspection PhpUnusedParameterInspection */
    ?MessageModel $messageModel = null,
    ?array $messageRaw = null,
    int $endFlag = -1
) use (&$result) {
    if ($endFlag === -1) {
        arsort($result, SORT_NUMERIC);

        echo "\tSite\t|\tLinks count\t\n";
        foreach ($result as $site => $count) {
            echo $site."\t|\t".$count."\n";
        }
    } elseif (!empty($messageRaw['message']) &&
        preg_match('/http[s]?:\/\/\w+([\.-]*)\/', $messageRaw['message'],
            $matches) && !empty($matches[1])) {
        $domain = $matches[1];
        $result[$domain] = !empty($result[$domain]) ?
            $result[$domain] + 1 : 1;
    }
};

$onGroupReady = static function (
    ?int $groupId = null,
    ?int $accessHash = null
) use ($timestampStart, $timestampEnd, $generator, $parsedLinks) {
    if (!$groupId) {
        Logger::log('parseChannelLinks', 'Group not found');

        return;
    }

    $client = new LinkParseScenario(
        new GroupId($groupId, $accessHash),
        $generator,
        new OptionalDateRange(
            $timestampStart,
            $timestampEnd
        ),
        $parsedLinks
    );
    $client->startActions();
};

/** @noinspection PhpUnhandledExceptionInspection */

```

```
$resolver = new GroupResolverScenario($request, $generator,
$onGroupReady);
/** @noinspection PhpUnhandledExceptionInspection */
$resolver->startActions();
```

Сценарий принимает на вход ссылку на канал, начало временного отрезка, конец временного отрезка. Задача сценария перебрать все публикации в канале на заданном отрезке времени и подсчитать количество упоминаний каждой ссылки. В случае обнаружения ошибок в ссылке вывести соответствующее сообщение об ошибке.

Для данного сценария будут вычислены две группы метрик: метрики размера программы и метрики Холстеда.

4.5.1 Метрики размера программы

Первой проведенной метрикой будет подсчет размера программы, где будет вычислено количество строк программного кода, а также число комментариев. Метрики будут найдены на основе выше приведенного кода. Результат представлен в таблице 7.

Таблица 7 – Метрики размера программы

Метрика	Количество
Физических строк кода	77
Логических строк кода	46
Строки-комментарии	18

Приведенные количественные значения дают возможность вычислить процент комментариев для рассматриваемого участка кода. Процент комментариев равен $(18/77) * 100 = 23$. Полученный результат свидетельствует о достаточности комментирования кода. Это полезно для облегчения понимания кода другими разработчиками и обеспечения его поддержки в будущем.

4.5.2 Метрики Холстеда

Метрики Холстеда – это набор метрик программного обеспечения, предложенных Маурисом Холстедом, которые используются для измерения различных характеристик программного кода. Они включают объем программы, сложность программы, уровень абстракции и усилия разработки. Метрики Холстеда помогают оценить сложность, трудоемкость и качество

программы, а также проводить сравнительный анализ различных версий или модулей программного кода [10].

Для вычисления метрик Холстеда необходимо измерить 4 характеристики программы

- число уникальных операторов;
- число уникальных операндов;
- общее число операторов;
- общее число операндов.

Для вычисления характеристик были составлены словарь операторов, которой отображен в таблице 8 и словарь операндов, который описан в таблице 9.

Таблица 8 – Словарь операторов

Оператор	Число операторов
if	5
+=	0
!=	1
.	35
()	11
{}	7
;	29
[]	19
=	36
+	1

Таблица 9 – Словарь операндов

Операнд	Число операндов
argsOrFalse	13
hasGroupId	2
hasLink	2
groupId	10
deepLink	2
timestampStart	6
timestampEnd	6
generator	5
request	4
result	7
parseLinks	3

Таким образом, число уникальных операторов программы $n1 = 10$, число уникальных операндов программы $n2 = 11$, общее число операторов в программе $N1 = 144$, общее число операндов в программе $N2 = 60$.

На основе метрик Холстеда можно вычислить целый ряд характеристик программы. Рассмотрим некоторые из них:

- словарь программы $n = n_1 + n_2$;
- длина программы $N = N_1 + N_2$;
- объем программы $V = N * \log_2 n$;
- сложность программы $D = (n_1/2) * (N_2/n_2)$;
- оценка интеллектуальных усилий $E = D * V$.

Рассчитаем словарь программы: $n = 10 + 11 = 21$.

Рассчитаем длину программы: $N = 144 + 60 = 204$.

Рассчитаем объем программы: $V = 204 * \log_2 21 = 204 * 4,3 = 877,2$

Рассчитаем сложность программы: $D = (10/2) * (60/11) = 5 * 5,45 = 27,25$.

Рассчитаем оценку интеллектуальных усилий: $E = 27,25 * 877,2 = 23903,7$.

Подсчитанные значения предоставляют информацию о том, что в проекте количество используемых переменных не велико, а значит данный проект будет проще поддерживать в будущем.

Также длина программы и ее объем показывают, что код является обширным, но при этом он разбит на логические функциональные блоки, которые легко масштабировать.

Сложность программы и оценка интеллектуальных усилий демонстрирует, что имеется множества высокоуровневых связей.

Полученные характеристики показывают, что программный код имеет достаточно высокий уровень качества и удобства для дальнейшей поддержки.

4.6 Планирование разработки и оценка бюджета

Планирование разработки и оценка бюджета – это две важные фазы любого проекта разработки программного обеспечения. Они позволяют определить порядок выполнения задач, распределить ресурсы и оценить затраты на проект.

Планирование разработки программного обеспечения – это ключевой этап в жизненном цикле разработки программного обеспечения (Software Development Life Cycle, SDLC). Он включает в себя определение требований к

проекту, оценку затрат и ресурсов, а также разработку подробного плана работы.

Существуют различные методологии планирования, такие как Agile, Waterfall и Scrum, каждая из которых имеет свои особенности. Однако, вне зависимости от выбранной методологии, планирование разработки обычно включает в себя следующие шаги.

Определение требований. Этот шаг включает в себя определение функциональных и нефункциональных требований к программному обеспечению. Это важный этап, который позволяет разработчикам понять, что именно должно быть реализовано [11].

Оценка. На основе требований проекта, разработчики оценивают необходимые для выполнения работы ресурсы. Оценка может включать в себя затраты на разработку, тестирование, поддержку и т. д.

Разработка плана. На основе определенных требований и оценок, разрабатывается план работы. Он включает в себя расписание выполнения задач, ресурсы, которые будут использоваться, и т. д.

Оценка бюджета важна для определения общих затрат на проект. Она может включать затраты на разработку, тестирование, поддержку, а также другие затраты, связанные с проектом, например, затраты на обучение персонала, лицензии на программное обеспечение и т. д.

Таким образом, планирование разработки и оценка бюджета – это важные этапы в жизненном цикле разработки программного обеспечения, которые помогают обеспечить успешное выполнение проекта [12].

При планировании разработки было выделено 22 задачи. Список задач с общей информацией (длительность, дата начала, дата окончания и др.) представлены на рисунке 14.

		Название задачи	Длительность	Начало	Окончание	Предшественник	Названия ресурсов
1		Начало проекта	0 дней	Пн 03.04.23	Пн 03.04.23		
2		Анализ требований	9 дней	Пн 03.04.23	Чт 13.04.23		
3		Изучение предметной области	5 дней	Пн 03.04.23	Пт 07.04.23	1	Компьютер[1];Программист
4		Анализ аналогичных приложений	4 дней	Пн 10.04.23	Чт 13.04.23	3	Компьютер[1];Программист
5		Анализ требований завершен	0 дней	Чт 13.04.23	Чт 13.04.23	4	
6		Выбор средств	5 дней	Пн 10.04.23	Пт 14.04.23		
7		Выбор программных средств	3 дней	Пн 10.04.23	Ср 12.04.23	3	Программист ;Компьютер[1]
8		Выбор методологии разработки	2 дней	Чт 13.04.23	Пт 14.04.23	7	Программист ;Компьютер[1]
9		Завершение выбора средств	0 дней	Пт 14.04.23	Пт 14.04.23	8	
10		Проектирование приложения	10 дней	Пн 17.04.23	Пт 28.04.23		
11		Проектирование пользовательского интерфейса	5 дней	Пн 17.04.23	Пт 21.04.23	9	Программист ;Компьютер[1]
12		Проектирование реализуемых функций	5 дней	Пн 24.04.23	Пт 28.04.23	11	Программист ;Компьютер[1]
13		Завершение проектирования приложения	0 дней	Пт 28.04.23	Пт 28.04.23	12	
14		Реализация приложения	21 дней	Пн 01.05.23	Пн 29.05.23		
15		Разработка пользовательского интерфейса	9 дней	Пн 01.05.23	Чт 11.05.23	13	Компьютер[1];Программист
16		Разработка реализуемых функций	12 дней	Пт 12.05.23	Пн 29.05.23	15	Компьютер[1];Программист
17		Завершение реализации приложения	0 дней	Пн 29.05.23	Пн 29.05.23	16	
18		Тестирование	2 дней	Вт 30.05.23	Ср 31.05.23		
19		Тестирование методом "черного ящика"	2 дней	Вт 30.05.23	Ср 31.05.23	17	Тестировщик;Компьютер[1]
20		Функциональное тестирование	2 дней	Вт 30.05.23	Ср 31.05.23	17	Компьютер[1];Тестировщик
21		Тестирование завершено	0 дней	Ср 31.05.23	Ср 31.05.23	20	
22		Конец проекта	0 дней	Ср 31.05.23	Ср 31.05.23	21	

Рисунок 14 – Список задач проектирования

Для того чтобы успешно выполнить проект, необходимо использовать определенные ресурсы. Список ресурсов, их тип, краткое название, максимальное количество единиц, стандартная ставка, сверхурочная ставка, назначение и некоторые другие параметры, не представляющие особой важности, представлены на рисунке 15.

Название ресурса	Тип	Краткое название	Макс. единиц	Стандартная ставка	Ставка :сверхурочны:	Затраты на исполыз.	Начисление	Базовый календарь
Программист	Трудовой	П	200%	р.42 000,00/мес	р.300,00/ч	р.0,00	Пропорциональное	Стандартный
Тестировщик	Трудовой	Т	200%	р.39 000,00/мес	р.200,00/ч	р.0,00	Пропорциональное	Стандартный
Компьютер	Материальный	К		р.0,00		р.0,00	Пропорциональное	
Интернет	Затраты	И					Пропорциональное	

Рисунок 15 – Список ресурсов проекта

После назначения задач была получена полноценная диаграмма Ганта. Диаграмма Ганта – это инструмент, который позволяет эффективно планировать, координировать и отслеживать конкретные задачи в рамках проекта. Она показывает все задачи, которые необходимо выполнить, а также время, которое потребуется для их выполнения.

Каждая задача представлена в виде горизонтальной полосы, длина которой соответствует времени, необходимому для выполнения этой задачи.

Задачи располагаются вертикально в том порядке, в котором они должны быть выполнены, с самыми ранними задачами вверху диаграммы.

Диаграмма Ганта также может показывать зависимости между задачами. Если выполнение одной задачи не может начаться, пока не будет выполнена другая задача, эти две задачи связаны линией на диаграмме.

Использование диаграммы Ганта позволило наглядно представить план разработки, определить критический путь и эффективно распределить ресурсы. Это обеспечивает реалистичность планирования и контроль над выполнением задач, что является критически важным для успешного завершения проекта [13].

На диаграмме наглядно видно связи между задачами, используемые ресурсы и даты начала и окончания (рисунок 16).

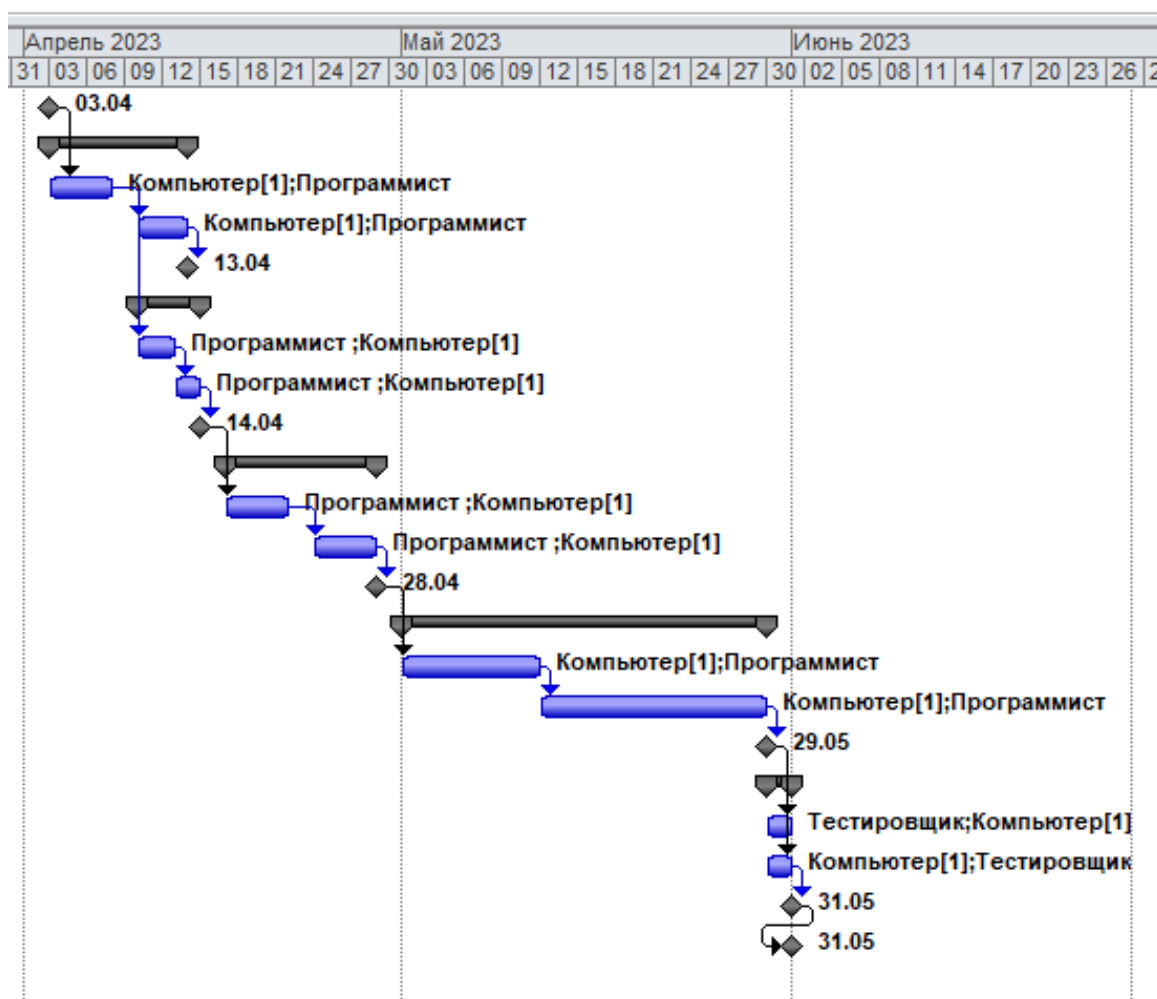


Рисунок 16 – Список ресурсов проекта

После составления проекта, определения дат и используемых ресурсов, был создан отчет, представленный на рисунке 17. Отчет отражает общую информацию о проекте, такую как дата начала, дата окончания, трудозатраты на разработку по часам, материальные затраты на ресурсы и общее количество ресурсов.

MSProj11			
на дату Ср 28.06.23			
Даты			
Начало:	Пн 03.04.23	Окончание:	Ср 31.05.23
Баз. начало:	НД	Баз. окончание:	НД
Факт. начало:	НД	Факт. окончание:	НД
Отклон. начала:	0 дней	Отклон. окончания:	0 дней
Длительность			
Текущий план:	43 дней	Оставшаяся:	43 дней
Базовый план:	0 дней?	Фактическая:	0 дней
Отклонение:	43 дней	Процент завершения	0%
Трудозатраты			
Текущий план:	392 ч	Оставшиеся:	392 ч
Базовый план:	0 ч	Фактические:	0 ч
Отклонение:	392 ч	Процент завершения	0%
Затраты			
Текущий план:	р.119 500,00	Оставшиеся:	р.119 500,00
Базовый план:	р.0,00	Фактические:	р.0,00
Отклонение:	р.119 500,00		
Состояние задачи		Состояние ресурса	
Еще не начатые задачи:	22	Трудовые ресурсы:	2
Выполняющиеся задачи:	0	Трудовые ресурсы с прев. дост	0
Завершенные задачи:	0	Материальные ресурсы:	1
Итого задач:	22	Итого ресурсов:	3

Рисунок 17 – Отчет по проекту

В результате оценки бюджета примерная стоимость разработки составила 119500 рублей. Для разработки потребуется 392 человеко-часа. В условиях, указанных при планировании, разработка займет 43 дня.

Заключение

Результатом выполнения выпускной квалификационной работы является разработка и реализация аналитической системы мониторинга Telegram-каналов. Это приложение позволяет оптимизировать просмотр публикаций в каналах Telegram, а также получать аналитику об интересующих пользователя Telegram-каналах.

В ходе выполнения этапа анализа требований была обоснована актуальность разрабатываемого приложения. Рассмотрены приложения-аналоги и осуществлен их обзор, выделены преимущества и недостатки. На основании этих критериев проведена оценка. Результатом стало получение интегральных оценок каждого аналога. Также выявлены функциональные и нефункциональные требования и проведен их анализ.

В процессе проектирования аналитической системы мониторинга Telegram-каналов была определена его архитектура. В процессе проектирования были определены технологии и инструментальные средства разработки, составлена диаграмма развертывания, а также спроектирован пользовательский интерфейс. Сервер был реализован с помощью PHPStorm, в разработке использован язык программирования PHP. Для реализации статических элементов страниц и шаблонов использован язык разметки HTML5 и CSS3. Часть динамических сценариев на страницах реализована при помощи JavaScript. Для изменения стилей страниц использованы каскадные таблицы стилей (CSS).

На этапе реализации была составлена диаграмма компонентов и диаграмма развертывания, которые реализованы в программе. Также был реализован весь спроектированный функционал.

Серверная и клиентская части приложения были протестированы с помощью метода черного ящика и функционального тестирования. В результате все тесты были пройдены.

В реализованной системе были проведены метрики размера программы и метрики Холстеда. Полученные характеристики показывают, что программный

код имеет достаточно высокий уровень качества и удобства для дальнейшей поддержки.

Также было осуществлено планирование разработки о оценки бюджета. Из результатов планирования следует, что проект займет 43 дня на реализацию, а расчетные затраты составят 119500 рублей, откуда можно сделать вывод об его экономической целесообразности.

В перспективе планируется использовать разработанную аналитическую систему в новостном издании для осуществления проверки и анализа публикуемых новостей и источников информации. Также будет осуществляться не только сопровождение и поддержка актуальности программы, но и расширение функциональности. Для этого необходимо существенно расширить функционал приложения, добавив новые возможности и способы взаимодействия с Telegram. Также планируется внедрить в приложение графическое отображение результатов аналитики, что поможет более оптимально и наглядно использовать систему.

Таким образом, поставленная по программной реализации аналитической системы Telegram–каналов задача выполнена в полной мере.

Список использованных источников

1. Аналитика Telegram-канала и источников подписчиков. [Электронный ресурс]. URL: <https://vc.ru/marketing/672351-analitika-telegram-kanala-i-istochnikov-odpischikov/> (дата обращения: 10.05.2023).
2. Рамбо, Дж. UML 2.0. Объектно-ориентированное моделирование и разработка / Дж. Рамбо, М. Блаха. – 2-е изд. – СПб. : Питер, 2007. – 544 с.
3. Telegram APIs. [Электронный ресурс]. URL: <https://core.telegram.org/api> (дата обращения: 13.05.2023).
4. Кролл, П. Rational Unified Process – это легко. Руководство по RUP для практиков: Пер. с англ. / М.: КУДИЦ-ОБРАЗ, 2004. – 432 с.
5. Введение в Telegram API. [Электронный ресурс]. URL: <https://machinelearningmastery.ru/introduction-to-the-telegram-api-b0cd220dbed2/> (дата обращения: 14.05.2023).
6. Буч, Г. Язык UML. Руководство пользователя, 2-е издание / Г. Буч, Дж. Рамбо, И. Якобсон; пер. с англ.: Н. Мухин. – М. : Издательский дом «ДМК Пресс», 2006. – 496 с.
7. Криспин, Г. Гибкое тестирование: практическое руководство для тестировщиков ПО и гибких команд.: Пер. с англ. – М.: ООО «И.Д. Вильямс», 2010. – 464 с.
8. Куликов, С. Тестирование программного обеспечения. Базовый курс, 3-е издание / С. Куликов. – М. : Издательство МГТУ им. Н. Э. Баумана, 2020. – 301 с.
9. Программный код и его метрики. [Электронный ресурс]. URL: <https://habr.com/ru/company/intel/blog/106082/> (дата обращения: 19.05.2023).
10. Ехлаков, Ю.П. Управление программными проектами. Стандарты, модели: Учебное пособие / Ю.П. Ехлаков. – СПб. : Лань, 2019. – 244 с.
11. Иванов, П.В. Управление проектами: Учебное пособие / П.В. Иванов. – Рн/Д: Феникс, 2018. – 102 с.
12. Гонтарева, И.В. Управление проектами / И.В. Гонтарева, Р.М. Нижегородцев, Д.А. Новиков. – М. : КД Либроком, 2018. – 384 с.

Приложение А
ЛИСТИНГ ПРОГРАММЫ
(рекомендуемое)

Серверная часть.

Листинг А.1 – Файл parseChannelLinks.php

```

<?php

/** @noinspection BypassedUrlValidationInspection */

declare(strict_types=1);

require_once __DIR__.'../../vendor/autoload.php';

use TelegramOSINT\Client\InfoObtainingClient\Models\GroupId;
use TelegramOSINT\Client\InfoObtainingClient\Models\MessageModel;
use TelegramOSINT\Logger\Logger;
use TelegramOSINT\Scenario\GroupResolverScenario;
use TelegramOSINT\Scenario\LinkParseScenario;
use TelegramOSINT\Scenario\Models\GroupRequest;
use TelegramOSINT\Scenario\Models\OptionalDateRange;
use TelegramOSINT\Scenario\ReusableClientGenerator;

$argsOrFalse = getopt('g:l:f:t:h', ['group-id:', 'deep-link:', 'timestamp-
from:', 'timestamp-to:', 'help']); //аргументы, принимаемые функцией
$hasGroupId = array_key_exists('g', $argsOrFalse) || array_key_exists('group-
id', $argsOrFalse);
$hasLink = array_key_exists('l', $argsOrFalse) || array_key_exists('deep-link',
$argsOrFalse);

if ($hasGroupId && $hasLink) {
    echo 'Group ID and Deep Link are mutually exclusive. Please specify only one
of them.'.PHP_EOL;          //вывод предупреждения при объявлении одновременно
ссылки и идентификатора

    exit(1);
}

$groupId = $argsOrFalse['g'] ?? $argsOrFalse['group-id'] ?? null;
//идентификатор канала
$deepLink = $argsOrFalse['l'] ?? $argsOrFalse['deep-link'] ?? '';
//ссылка на канал
$timestampStart = $argsOrFalse['f'] ?? $argsOrFalse['timestamp-from'] ?? null;
//начало временного отрезка
$timestampEnd = $argsOrFalse['t'] ?? $argsOrFalse['timestamp-to'] ?? null;
//конец временного отрезка

$timestampStart = $timestampStart === null ? null : (int) $timestampStart;
$timestampEnd = $timestampEnd === null ? null : (int) $timestampEnd;

$generator = new ReusableClientGenerator(); //генератор клиента telegram
$request = $groupId !== null
    ? GroupRequest::ofGroupId((int) $groupId) //запрос клиента к каналу по id
    : GroupRequest::ofUserName($deepLink);    //запрос клиента по ссылке

$result = [];
$parseLinks = static function ( //функция подсчета
ссылок в канале
    /** @noinspection PhpUnusedParameterInspection */
    ?MessageModel $messageModel = null, //модель сообщения
    ?array $messageRaw = null,
    int $endFlag = -1
) use (&$result) {
    if ($endFlag === -1) {
        arsort($result, SORT_NUMERIC);

        echo "\tSite\t|\tLinks count\t\n";
    }
}

```

```

        foreach ($result as $site => $count) {
            echo $site."\t|\t".$count."\n";           //вывод количества
упоминаний ссылки
        }
    } elseif (!empty($messageRaw['message']) &&
preg_match('/http[s]?:\/\/\:\/\/([\w.\-_]*)\/', $messageRaw['message'], $matches) &&
!empty($matches[1])) {
        $domain = $matches[1];
        $result[$domain] = !empty($result[$domain]) ? $result[$domain] + 1 : 1;
    }
};

$onGroupReady = static function (
    ?int $groupId = null,
    ?int $accessHash = null
) use ($timestampStart, $timestampEnd, $generator, $parseLinks) {
    if (!$groupId) {
        Logger::log('parseChannelLinks', 'Group not found');           //вывод
ошибки, если канал не найден

        return;
    }

    $client = new LinkParseScenario(
        new GroupId($groupId, $accessHash),           //инициализация идентификатора
канала
        $generator,
        new OptionalDateRange(
            $timestampStart,
            $timestampEnd
        ),
        $parseLinks
    );
    $client->startActions();
};

/** @noinspection PhpUnhandledExceptionInspection */
$resolver = new GroupResolverScenario($request, $generator, $onGroupReady);
/** @noinspection PhpUnhandledExceptionInspection */
$resolver->startActions();

```

Листинг А.2 – Файл parseChannelMessages.php

```

<?php

declare(strict_types=1);

require_once __DIR__.'../../vendor/autoload.php';

use TelegramOSINT\Client\InfoObtainingClient\Models\GroupId;
use TelegramOSINT\Logger\Logger;
use TelegramOSINT\Scenario\GroupMessagesScenario;
use TelegramOSINT\Scenario\GroupResolverScenario;
use TelegramOSINT\Scenario\Models\GroupRequest;
use TelegramOSINT\Scenario\Models\OptionalDateRange;
use TelegramOSINT\Scenario\ReusableClientGenerator;

$argsOrFalse = getopt('g:l:f:t:u:h', ['group-id:', 'deep-link:', 'timestamp-
from:', 'timestamp-to:', 'user:', 'help']);
$hasGroupId = array_key_exists('g', $argsOrFalse) || array_key_exists('group-
id', $argsOrFalse);
$hasLink = array_key_exists('l', $argsOrFalse) || array_key_exists('deep-link',
$argsOrFalse);
if ($argsOrFalse === false

```

```

    || (array_key_exists('h', $argsOrFalse) || array_key_exists('help',
    $argsOrFalse))
    || (!$hasGroupId && !$hasLink)
) {
    echo <<<'EOT'
Usage:
    php parseGroupMessages.php -g groupID | -l link
                                [-f timestampFrom] [-t timestampTo] [-u username]
    php parseGroupMessages.php --group-id groupID | --deep-link link
                                [--timestamp-from timestampFrom] [--timestamp-to
timestampTo] [--user username]

    Note: Group ID and Deep Link are mutually exclusive. Please specify only one
of them.

    -g, --group-id                Group identifier.
    -l, --deep-link               Deep link (e.g. https://t.me/vityapelevin).
    -f, --timestamp-from          Optional start timestamp.
    -t, --timestamp-to            Optional end timestamp.
    -u, --user                    Optional user name.
    -h, --help                    Display this help message.

EOT;
    exit(1);
}

if ($hasGroupId && $hasLink) {
    echo 'Group ID and Deep Link are mutually exclusive. Please specify only one
of them.'.PHP_EOL;

    exit(1);
}

$groupId = $argsOrFalse['g'] ?? $argsOrFalse['group-id'] ?? null;
$deepLink = $argsOrFalse['l'] ?? $argsOrFalse['deep-link'] ?? '';
$username = $argsOrFalse['u'] ?? $argsOrFalse['user'] ?? null;
$timestampStart = $argsOrFalse['f'] ?? $argsOrFalse['timestamp-from'] ?? null;
$timestampEnd = $argsOrFalse['t'] ?? $argsOrFalse['timestamp-to'] ?? null;

$timestampStart = $timestampStart === null ? null : (int) $timestampStart;
$timestampEnd = $timestampEnd === null ? null : (int) $timestampEnd;

$generator = new ReusableClientGenerator();
$request = $groupId !== null
    ? GroupRequest::ofGroupId((int) $groupId)
    : GroupRequest::ofUserName($deepLink);

/**
 * @param int|null $groupId
 * @param int|null $accessHash
 */
$onGroupReady = static function (
    ?int $groupId,
    ?int $accessHash = null
) use ($timestampStart, $timestampEnd, $username, $generator): void {
    if (!$groupId || !$accessHash) {
        Logger::log('parseGroupMessages', 'Group not found');

        return;
    }

    /** @var int $accessHash */
    $client = new GroupMessagesScenario(
        new GroupId($groupId, $accessHash),
        $generator,

```

```

        new OptionalDateRange(
            $timestampStart,
            $timestampEnd,
        ),
        null,
        $username,
        100,
        true
    );

    $client->startActions(false);
};

Logger::log(__FILE__, 'starting group resolver for '.$request);
/** @noinspection PhpUnhandledExceptionInspection */
$resolver = new GroupResolverScenario($request, $generator, $onGroupReady);
/** @noinspection PhpUnhandledExceptionInspection */
$resolver->startActions();

```

Листинг А.3 – Файл searchChannelMessages.php

```

<?php

declare(strict_types=1);

require_once __DIR__.'../../vendor/autoload.php';

use TelegramOSINT\Scenario\GroupMembersScenario;

$argsOrFalse = getopt('g:l:u:h', ['group-id:', 'deep-link:', 'username:',
    'help']);

if ($argsOrFalse === false
    || (array_key_exists('h', $argsOrFalse) || array_key_exists('help',
    $argsOrFalse))
) {
    echo <<<'EOT'

Usage:

php parseGroupMembers.php [-g groupID | -l link]
php parseGroupMembers.php [--group-id groupID | --deep-link link]

    Note: Group ID and Deep Link are mutually exclusive. Please specify only one
    of them.

    -g, --group-id                Optional group identifier.
    -u, --username                Search by username.
    -l, --deep-link                Optional deep link (e.g.
    https://t.me/vityapelevin).
    -h, --help                    Display this help message.

EOT;

    exit(1);

```

```

}

$groupId = $argsOrFalse['g'] ?? $argsOrFalse['group-id'] ?? null;
$deepLink = $argsOrFalse['l'] ?? $argsOrFalse['deep-link'] ?? '';
$username = $argsOrFalse['u'] ?? $argsOrFalse['username'] ?? null;
if ($username) {
    echo "searching usernames like: $username".PHP_EOL;
}
if ($groupId !== null && $deepLink !== '') {
    echo 'Group ID and Deep Link are mutually exclusive. Please specify only one
of them.'.PHP_EOL;

    exit(1);
}

/** @noinspection PhpUnhandledExceptionInspection */
$client = new GroupMembersScenario(
    null,
    null,
    null,
    100,
    $username
);
if ($groupId !== null) {
    $client->setGroupId((int) $groupId);
} elseif ($deepLink !== '') {
    $client->setDeepLink($deepLink);
}
/* @noinspection PhpUnhandledExceptionInspection */
$client->startActions();

```

Листинг А.4 – Файл countChannelMessages.h

```

<?php

declare(strict_types=1);

use TelegramOSINT\Scenario\CommonChatsScenario;
use TelegramOSINT\Scenario\ReusableClientGenerator;

require_once __DIR__.'../../vendor/autoload.php';

$argsOrFalse = getopt('n:h', ['number:', 'help']);
if ($argsOrFalse === false
    || (array_key_exists('h', $argsOrFalse) || array_key_exists('help',
$argsOrFalse))
    || (!array_key_exists('n', $argsOrFalse) && !array_key_exists('number',
$argsOrFalse))
) {
    echo <<<'EOT'

```

```

Usage:
  php commonChats.php -n number
  php commonChats.php --number number

  -n, --number          Phone number (e.g. 1234567890).
  -h, --help            Display this help message.

EOT;
    exit(1);
}

$phone = $argsOrFalse['n'] ?? $argsOrFalse['number'];

$chatMap = [
    'twochannel'          => ['политика', 'развлечения'],
    'durclub'            => ['развлечения', 'игры'],
    'ateo_chat'          => ['развлечения', 'политика'],
    'chop_chop_xxx'      => ['развлечения'],
    'flibustafreebookbot_new' => ['книги'],
    'znakomstva_chats'   => ['знакомства'],
    'phuketrusa'         => ['путешествия', 'пхукет'],
];

$callback = static function ($interests) use ($phone) {
    arsort($interests);
    echo 'Phone number: '.$phone."\n";
    echo "Interest\t|\tWeight\n";
    foreach ($interests as $title => $rating) {
        echo $title."\t|\t".$rating."\n";
    }
};

$generator = new ReusableClientGenerator();
/** @noinspection PhpUnhandledExceptionInspection */
$scenario = new CommonChatsScenario(
    $generator,
    $chatMap,
    $phone,
    $callback
);
/** @noinspection PhpUnhandledExceptionInspection */
$scenario->startActions();

```

Листинг A.5 – Файл getChannelDescription.php

```

<?php

declare(strict_types=1);

use TelegramOSINT\Client\InfoObtainingClient\Models\UserInfoModel;
use TelegramOSINT\LibConfig;
use TelegramOSINT\Scenario\ClientGenerator;
use TelegramOSINT\Scenario\UserContactsScenario;
use TelegramOSINT\Tools\Proxy;

require_once __DIR__.'../../vendor/autoload.php';

// here we get contact list and get contact online status
// avatars are saved to current directory

$argsOrFalse = getopt('n:u:hwp:', ['numbers:', 'users:', 'help', 'photo',
'proxy:']);
if ($argsOrFalse === false
    || (array_key_exists('h', $argsOrFalse) || array_key_exists('help',
$argsOrFalse)))

```

```

    || (!array_key_exists('n', $argsOrFalse) && !array_key_exists('numbers',
$argsOrFalse))
        && (!array_key_exists('u', $argsOrFalse) && !array_key_exists('users',
$argsOrFalse)))
    ) {
        echo <<<'EOT'
Usage:
    php parseNumbers.php -n numbers [ -u users ]
    php parseNumbers.php --numbers numbers [ --photo ]
    php parseNumbers.php --users users [ --proxy proxyString ]

    -n, --numbers           Comma separated phone number list (e.g.
79061231231,79061231232).
    -u, --users             Comma separated username list (e.g. aaa,bbb).
    -p, --proxy             Proxy to use.
    -w, --photo             Download photo.
    -h, --help              Display this help message.

EOT;
        exit(1);
    }

$numbers = array_filter(explode(',', $argsOrFalse['n'] ??
$argsOrFalse['numbers'] ?? ''));
$users = array_filter(explode(',', $argsOrFalse['u'] ?? $argsOrFalse['users'] ??
''));

$onComplete = static function (UserInfoModel $model) {
    $photo_file = '';
    if ($model->photo) {
        $photo_file = $model->phone.'.'.$model->photo->format;
        file_put_contents(
            $photo_file,
            $model->photo->bytes
        );
    }
    echo implode("\t|\t", [
        $model->phone,
        $model->username,
        $model->firstName,
        $model->lastName,
        $photo_file,
        $model->bio,
        $model->commonChatsCount,
        $model->langCode,
        '',
    ]);

    if ($model->status->was_online) {
        echo date('Y-m-d H:i:s', $model->status->was_online).PHP_EOL;
    } elseif ($model->status->is_hidden) {
        echo 'Hidden'.PHP_EOL;
    } elseif ($model->status->is_online) {
        echo 'Online'.PHP_EOL;
    } else {
        echo "\n";
    }
};

$withPhoto = isset($argsOrFalse['w']) || isset($argsOrFalse['photo']);
if ($withPhoto) {
    echo 'parsing with photos'.PHP_EOL;
}

$proxyStr = $argsOrFalse['p'] ?? $argsOrFalse['proxy'] ?? null;

```

```

$separator = "\t|\t";
echo implode($separator, [
    'Phone',
    'Username',
    'First name',
    'Last name',
    'Photo',
    'About',
    'Common chats',
    'Lang',
    'Was online',
]);
.PHP_EOL.PHP_EOL;
/** @noinspection PhpUnhandledExceptionInspection */
$client = new UserContactsScenario(
    $numbers,
    $users,
    $onComplete,
    new ClientGenerator(LibConfig::ENV_AUTHKEY, $proxyStr ? new Proxy($proxyStr)
: null),
    $withPhoto,
    false
);
/* @noinspection PhpUnhandledExceptionInspection */
$client->startActions();

```

Клиентская часть.

Листинг А.6 – Файл Style.css

```

.rectangle-44 {
    width: 877px;
    height: 53px;
    padding: 8px 8px 8px 8px;
    background: #424242;
}
.rectangle-46 {
    width: 156px;
    height: 53px;
    padding: 8px 8px 8px 8px;
    background: #000000;
}
.-2 {
    width: 125px;
    height: 25px;
    color: #898989;
    font-family: "Helvetica";
    font-weight: 400;
    font-size: 18px;
    text-align: left;
}
.rectangle-47 {
    width: 877px;
    height: 652px;
    padding: 8px 8px 8px 8px;
    background: #ffffff;
    border-color: #dddddd;
    border-width: 1px;
    border-style: solid;
    border-radius: 3px 3px 3px 3px;
}
.rectangle-49 {
    width: 261px;
    height: 38px;
}

```

```

padding: 8px 8px 8px 8px;
background: #eeeeee;
border-color: #dddddd;
border-width: 1px;
border-style: solid;
border-radius: 3px 3px 0px 0px;
}
.-3 {
width: 213px;
height: 23px;
color: #999999;
font-family: "Helvetica";
font-weight: 700;
font-size: 16px;
text-align: left;
}
.rectangle-45 {
width: 114px;
height: 53px;
padding: 8px 8px 8px 8px;
background: #000000;
}
.-2 {
width: 80px;
height: 25px;
color: #898989;
font-family: "Helvetica";
font-weight: 400;
font-size: 18px;
text-align: left;
}
.rectangle-48 {
width: 261px;
height: 592px;
padding: 8px 8px 8px 8px;
background: #ffffff;
border-color: #dddddd;
border-width: 1px;
border-style: solid;
border-radius: 3px 3px 3px 3px;
}
.rectangle-56 {
width: 172px;
height: 32px;
padding: 8px 8px 8px 8px;
background: #428bca;
color: #929292;
border-color: #cccccc;
border-width: 1px;
border-style: solid;
border-radius: 3px 0px 0px 3px;
font-family: "Helvetica";
font-weight: 400;
font-size: 14px;
line-height: 1.3;
text-align: left;
}
.rectangle-57 {
width: 75px;
height: 32px;
padding: 8px 8px 8px 8px;
background: #428bca;
border-color: #cccccc;
border-width: 1px;
border-style: solid;

```

```

        border-radius: 0px 3px 3px 0px;
    }
    .rectangle-58 {
        width: 583px;
        height: 592px;
        padding: 8px 8px 8px 8px;
        background: #ffffff;
        border-color: #dddddd;
        border-width: 1px;
        border-style: solid;
        border-radius: 3px 3px 3px 3px;
    }
    .rectangle-59 {
        width: 583px;
        height: 38px;
        padding: 8px 8px 8px 8px;
        background: #eeeeee;
        border-color: #dddddd;
        border-width: 1px;
        border-style: solid;
        border-radius: 3px 3px 0px 0px;
    }
    .-5 {
        width: 65px;
        height: 25px;
        color: #000000;
        font-family: "Helvetica";
        font-weight: 400;
        font-size: 14px;
        text-align: center;
    }
    .-3 {
        width: 516px;
        height: 26px;
        color: #999999;
        font-family: "Helvetica";
        font-weight: 700;
        font-size: 16px;
        text-align: left;
    }
    .rectangle-60 {
        width: 477px;
        height: 32px;
        padding: 8px 8px 8px 8px;
        background: #428bca;
        border-color: #cccccc;
        border-width: 1px;
        border-style: solid;
        border-radius: 3px 0px 0px 3px;
    }
    .- {
        width: 468px;
        height: 19px;
        color: #7a7a7a;
        font-family: "Helvetica";
        font-weight: 400;
        font-size: 14px;
        text-align: left;
    }
    .rectangle-43 {
        width: 75px;
        height: 32px;
        padding: 8px 8px 8px 8px;
        background: #428bca;
        border-color: #cccccc;

```

```

        border-width: 1px;
        border-style: solid;
        border-radius: 0px 3px 3px 0px;
    }
    .-1 {
        width: 48px;
        height: 25px;
        color: #000000;
        font-family: "Helvetica";
        font-weight: 400;
        font-size: 14px;
        text-align: left;
    }
    .rectangle-69 {
        width: 833px;
        height: 251px;
        padding: 8px 8px 8px 8px;
        background: #ffffff;
        color: #000000;
        border-color: #dddddd;
        border-width: 1px;
        border-style: solid;
        border-radius: 3px 3px 3px 3px;
        font-family: "Helvetica";
        font-weight: 400;
        font-size: 14px;
        line-height: 1.3;
        text-align: left;
    }
    .text-placeholder-5 {
        width: 388px;
        height: 228px;
        color: #000000;
        font-family: "Helvetica";
        font-weight: bold;
        font-size: 16px;
        text-decoration: underline;
        line-height: 1.3;
        text-align: left;
    }
    .rectangle-70 {
        width: 833px;
        height: 38px;
        padding: 8px 8px 8px 8px;
        background: #eeeeee;
        border-color: #dddddd;
        border-width: 1px;
        border-style: solid;
        border-radius: 3px 3px 0px 0px;
    }
    .-1 {
        width: 752px;
        height: 18px;
        color: #999999;
        font-family: "Helvetica";
        font-weight: 400;
        font-size: 16px;
        text-align: left;
    }
    .button-6 {
        width: 39px;
        height: 39px;
        padding: 0px 10px 0px 10px;
        background: #ebebeb;
        color: #232323;
    }

```

```

        border-color: #232323;
        border-width: 1px;
        border-style: solid;
        border-radius: 3px 3px 3px 3px;
        font-family: "Helvetica";
        font-weight: 400;
        font-size: 12px;
        text-align: center;
    }
    .rectangle-78 {
        width: 120px;
        height: 32px;
        padding: 8px 8px 8px 8px;
        background: #428bca;
        border-color: #cccccc;
        border-width: 1px;
        border-style: solid;
        border-radius: 0px 3px 3px 0px;
    }
    .-6 {
        width: 104px;
        height: 25px;
        color: #000000;
        font-family: "Helvetica";
        font-weight: 400;
        font-size: 14px;
        text-align: center;
    }
    .date-picker-8 {
        width: 117px;
        height: 39px;
        padding: 4px 8px 4px 8px;
        background: #ebebeb;
        color: #7a7a7a;
        border-color: #232323;
        border-width: 1px;
        border-style: solid;
        border-radius: 3px 3px 3px 3px;
        font-family: "Helvetica";
        font-weight: 400;
        font-size: 14px;
        text-align: left;
    }
    .rectangle-65 {
        width: 400px;
        height: 209px;
        padding: 8px 8px 8px 8px;
        background: #ffffff;
        color: #000000;
        border-color: #dddddd;
        border-width: 1px;
        border-style: solid;
        border-radius: 3px 3px 3px 3px;
        font-family: "Helvetica";
        font-weight: 400;
        font-size: 14px;
        line-height: 1.3;
        text-align: left;
    }
    .rectangle-67 {
        width: 419px;
        height: 209px;
        padding: 8px 8px 8px 8px;
        background: #ffffff;
        color: #000000;

```

```

border-color: #dddddd;
border-width: 1px;
border-style: solid;
border-radius: 3px 3px 3px 3px;
font-family: "Helvetica";
font-weight: 400;
font-size: 14px;
line-height: 1.3;
text-align: left;
}
.text-placeholder-7 {
width: 393px;
height: 139px;
color: #000000;
font-family: "Helvetica";
font-weight: bold;
font-size: 16px;
text-decoration: underline;
line-height: 1.3;
text-align: left;
}
.-3 {
width: 64px;
height: 32px;
color: #232323;
font-family: "Helvetica";
font-weight: 400;
font-size: 16px;
text-align: left;
}
.rectangle-77 {
width: 276px;
height: 32px;
padding: 8px 8px 8px 8px;
background: #428bca;
color: #929292;
border-color: #cccccc;
border-width: 1px;
border-style: solid;
border-radius: 3px 0px 0px 3px;
font-family: "Helvetica";
font-weight: 400;
font-size: 14px;
line-height: 1.3;
text-align: left;
}
.date-picker-7 {
width: 118px;
height: 39px;
padding: 4px 8px 4px 8px;
background: #ebebeb;
color: #7a7a7a;
border-color: #232323;
border-width: 1px;
border-style: solid;
border-radius: 3px 3px 3px 3px;
font-family: "Helvetica";
font-weight: 400;
font-size: 14px;
text-align: left;
}
.-3 {
width: 211px;
height: 32px;
color: #232323;

```

```

        font-family: "Helvetica";
        font-weight: 400;
        font-size: 16px;
        text-align: left;
    }
    .date-picker-6 {
        width: 118px;
        height: 32px;
        padding: 4px 8px 4px 8px;
        background: #ebebeb;
        color: #7a7a7a;
        border-color: #232323;
        border-width: 1px;
        border-style: solid;
        border-radius: 3px 3px 3px 3px;
        font-family: "Helvetica";
        font-weight: 400;
        font-size: 14px;
        text-align: left;
    }
    .date-picker-5 {
        width: 118px;
        height: 32px;
        padding: 4px 8px 4px 8px;
        background: #ebebeb;
        color: #7a7a7a;
        border-color: #232323;
        border-width: 1px;
        border-style: solid;
        border-radius: 3px 3px 3px 3px;
        font-family: "Helvetica";
        font-weight: 400;
        font-size: 14px;
        text-align: left;
    }
    .button-5 {
        width: 40px;
        height: 32px;
        padding: 0px 10px 0px 10px;
        background: #ebebeb;
        color: #232323;
        border-color: #232323;
        border-width: 1px;
        border-style: solid;
        border-radius: 3px 3px 3px 3px;
        font-family: "Helvetica";
        font-weight: 400;
        font-size: 12px;
        text-align: center;
    }
    .-2 {
        width: 106px;
        height: 26px;
        color: #232323;
        font-family: "Helvetica";
        font-weight: 400;
        font-size: 16px;
        text-align: left;
    }
    .-2 {
        width: 32px;
        height: 26px;
        color: #232323;
        font-family: "Helvetica";
        font-weight: 400;
    }

```

```

        font-size: 16px;
        text-align: left;
    }
    .rectangle-66 {
        width: 400px;
        height: 31px;
        padding: 8px 8px 8px 8px;
        background: #eeeeee;
        border-color: #dddddd;
        border-width: 1px;
        border-style: solid;
        border-radius: 3px 3px 0px 0px;
    }
    .-1 {
        width: 350px;
        height: 15px;
        color: #999999;
        font-family: "Helvetica";
        font-weight: 400;
        font-size: 14px;
        text-align: left;
    }
    .rectangle-68 {
        width: 419px;
        height: 31px;
        padding: 8px 8px 8px 8px;
        background: #eeeeee;
        border-color: #dddddd;
        border-width: 1px;
        border-style: solid;
        border-radius: 3px 3px 0px 0px;
    }
    .-1 {
        width: 378px;
        height: 15px;
        color: #999999;
        font-family: "Helvetica";
        font-weight: 400;
        font-size: 14px;
        text-align: left;
    }
}

```

Листинг А.7 – Файл public.html

```

<!doctype html>
<html lang="en" data-bs-theme="auto">
  <head><script src="/docs/5.3/assets/js/color-modes.js"></script>

  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <meta name="generator" content="Hugo 0.112.5">
  <title>Telegram analytics</title>

  <link href="/docs/5.3/dist/css/bootstrap.min.css" rel="stylesheet"
  integrity="sha384-
  9ndCyUaIbzAi2FUVXJi0CjmcCapSm07SnpJef0486qhlLnuZ2cdeRhO02iuK6FUUVM"
  crossorigin="anonymous">

  <!-- Favicons -->
  <link rel="apple-touch-icon" href="/docs/5.3/assets/img/favicons/apple-touch-
  icon.png" sizes="180x180">
  <link rel="icon" href="/docs/5.3/assets/img/favicons/favicon-32x32.png"
  sizes="32x32" type="image/png">
  <link rel="icon" href="/docs/5.3/assets/img/favicons/favicon-16x16.png"
  sizes="16x16" type="image/png">

```

```

<link rel="manifest" href="/docs/5.3/assets/img/favicons/manifest.json">
<link rel="mask-icon" href="/docs/5.3/assets/img/favicons/safari-pinned-tab.svg"
color="#712cf9">
<link rel="icon" href="/docs/5.3/assets/img/favicons/favicon.ico">
<meta name="theme-color" content="#712cf9">
  <!-- Custom styles for this template -->
  <link href="https://cdn.jsdelivr.net/npm/bootstrap-
icons@1.10.3/font/bootstrap-icons.css" rel="stylesheet">
  <!-- Custom styles for this template -->
  <link href="dashboard.css" rel="stylesheet">
</head>
<body>
  <svg xmlns="http://www.w3.org/2000/svg" style="display: none;">
    <symbol id="check2" viewBox="0 0 16 16">
      <path d="M13.854 3.646a.5.5 0 0 1 0 .708l-7 7a.5.5 0 0 1-.708 0l-3.5-
3.5a.5.5 0 1 1 .708-.708L6.5 10.293l6.646-6.647a.5.5 0 0 1 .708 0z"/>
    </symbol>
    <symbol id="circle-half" viewBox="0 0 16 16">
      <path d="M8 15A7 7 0 1 0 8 1v14zm0 1A8 8 0 1 1 8 0a8 8 0 0 1 0 16z"/>
    </symbol>
    <symbol id="moon-stars-fill" viewBox="0 0 16 16">
      <path d="M6 .278a.768.768 0 0 1 .08.858 7.208 7.208 0 0 0-.878 3.46c0
4.021 3.278 7.277 7.318 7.277.527 0 1.04-.055 1.533-.16a.787.787 0 0 1
.81.316.733.733 0 0 1-.031.893A8.349 8.349 0 0 1 8.344 16C3.734 16 0 12.286 0
7.71 0 4.266 2.114 1.312 5.124.06A.752.752 0 0 1 6 .278z"/>
      <path d="M10.794 3.148a.217.217 0 0 1 .412 0l.387 1.162c.173.518.579.924
1.097 1.097l1.162.387a.217.217 0 0 1 0 .412l-1.162.387a1.734 1.734 0 0 0-1.097
1.097l-.387 1.162a.217.217 0 0 1-.412 0l-.387-1.162A1.734 1.734 0 0 0 9.31
6.593l-1.162-.387a.217.217 0 0 1 0-.412l1.162-.387a1.734 1.734 0 0 0 1.097-
1.097l.387-1.162zM13.863.099a.145.145 0 0 1 .274
0l.258.774c.115.346.386.617.732.732l.774.258a.145.145 0 0 1 0 .274l-
.774.258a1.156 1.156 0 0 0-.732.732l-.774-.258a.145.145 0 0 1 0-.274l.774-
.774a1.156 1.156 0 0 0-.732-.732l-.774-.258a.145.145 0 0 1 0-.274l.774-
.258c.346-.115.617-.386.732-.732L13.863.1z"/>
    </symbol>
    <symbol id="sun-fill" viewBox="0 0 16 16">
      <path d="M8 12a4 4 0 1 0 0-8 4 4 0 0 0 8 0a.5.5 0 0 1 .5.5v2a.5.5 0
0 1-1 0v-2A.5.5 0 0 1 8 0zm0 13a.5.5 0 0 1 .5.5v2a.5.5 0 0 1-1 0v-2A.5.5 0 0 1 8
13zm8-5a.5.5 0 0 1-.5.5h-2a.5.5 0 0 1 0-1h2a.5.5 0 0 1 .5.5zM3 8a.5.5 0 0 1-
.5.5h-2a.5.5 0 0 1 0-1h2a.5.5 0 0 1 3 8zm10.657-5.657a.5.5 0 0 1 0 .707l-1.414
1.415a.5.5 0 1 1-.707-.708l1.414-1.414a.5.5 0 0 1 .707 0zm-9.193 9.193a.5.5 0 0
1 0 .707L3.05 13.657a.5.5 0 0 1-.707-.707l1.414-1.414a.5.5 0 0 1 .707 0zm9.193
2.121a.5.5 0 0 1-.707 0l-1.414-1.414a.5.5 0 0 1 .707-.707l1.414 1.414a.5.5 0 0 1
0 .707zM4.464 4.465a.5.5 0 0 1-.707 0L2.343 3.05a.5.5 0 1 1 .707-.707l1.414
1.414a.5.5 0 0 1 0 .708z"/>
    </symbol>
  </svg>

  <div class="dropdown position-fixed bottom-0 end-0 mb-3 me-3 bd-mode-
toggle">
    <button class="btn btn-bd-primary py-2 dropdown-toggle d-flex align-items-
center"
      id="bd-theme"
      type="button"
      aria-expanded="false"
      data-bs-toggle="dropdown"
      aria-label="Toggle theme (auto)">
      <svg class="bi my-1 theme-icon-active" width="1em" height="1em"><use
href="#circle-half"></use></svg>
      <span class="visually-hidden" id="bd-theme-text">Toggle theme</span>
    </button>
    <ul class="dropdown-menu dropdown-menu-end shadow" aria-labelledby="bd-
theme-text">
      <li>

```

```

        <button type="button" class="dropdown-item d-flex align-items-center"
data-bs-theme-value="light" aria-pressed="false">
        <svg class="bi me-2 opacity-50 theme-icon" width="1em"
height="1em"><use href="#sun-fill"></use></svg>
        Light
        <svg class="bi ms-auto d-none" width="1em" height="1em"><use
href="#check2"></use></svg>
        </button>
    </li>
    <li>
        <button type="button" class="dropdown-item d-flex align-items-center"
data-bs-theme-value="dark" aria-pressed="false">
        <svg class="bi me-2 opacity-50 theme-icon" width="1em"
height="1em"><use href="#moon-stars-fill"></use></svg>
        Dark
        <svg class="bi ms-auto d-none" width="1em" height="1em"><use
href="#check2"></use></svg>
        </button>
    </li>
    <li>
        <button type="button" class="dropdown-item d-flex align-items-center
active" data-bs-theme-value="auto" aria-pressed="true">
        <svg class="bi me-2 opacity-50 theme-icon" width="1em"
height="1em"><use href="#circle-half"></use></svg>
        Auto
        <svg class="bi ms-auto d-none" width="1em" height="1em"><use
href="#check2"></use></svg>
        </button>
    </li>
</ul>
</div>

<header class="navbar sticky-top bg-dark flex-md-nowrap p-0 shadow" data-bs-
theme="dark">
    <a class="navbar-brand col-md-3 col-lg-2 me-0 px-3 fs-6 text-white"
href="#">Публикации</a>
    <a class="navbar-brand col-md-3 col-lg-2 me-0 px-3 fs-6 text-white"
href="#">Каналы</a>

    <ul class="navbar-nav flex-row d-md-none">
        <li class="nav-item text-nowrap">
            <button class="nav-link px-3 text-white" type="button" data-bs-
toggle="collapse" data-bs-target="#navbarSearch" aria-controls="navbarSearch"
aria-expanded="false" aria-label="Toggle search">
                <svg class="bi"><use xlink:href="#search"/></svg>
            </button>
        </li>
        <li class="nav-item text-nowrap">
            <button class="nav-link px-3 text-white" type="button" data-bs-
toggle="offcanvas" data-bs-target="#sidebarMenu" aria-controls="sidebarMenu"
aria-expanded="false" aria-label="Toggle navigation">
                <svg class="bi"><use xlink:href="#list"/></svg>
            </button>
        </li>
    </ul>

    <div id="navbarSearch" class="navbar-search w-100 collapse">
        <input class="form-control w-100 rounded-0 border-0" type="text"
placeholder="Search" aria-label="Search">
    </div>
</header>

<div class="container-fluid">
    <div class="row">

```



```

tx.textContent = this.text;
if(this.type === 'radio' || this.type === 'checkbox'){
    var ans = document.createElement('div');
    field.appendChild(ans);
    for(var i = 0; i < this.answerList.length; i++){
        var point = document.createElement('input');
        ans.appendChild(point);
        point.type = this.type;
        point.name = this.questionId;
        point.value = this.answerList[i];
        point.id = 'ans' + pointcount;
        ans.innerHTML += '<label for="ans' + (pointcount++) + '">' +
this.answerList[i] + '</label><br>';
    }
    for(var i = 0; i < this.answerList.length; i++){
        var point = document.getElementById('ans' + i);
        point.onChange = function(){
            if(this.checked){
                qList[currentQuestion].giveAnswer(this.value);
                numcard.classList.add('num_answered');
            }
            else
                qList[currentQuestion].cancelAnswer(this.value);
            if(qList[currentQuestion].getGiven().length === 0)

document.getElementById(qList[currentQuestion].getId()).classList.remove('num_an
swered');

        };
        if(this.type === 'radio' && this.givenAnswer === point.value)
            point.checked = true;
        if(this.type === 'checkbox')
            for(var j = 0; j < this.givenAnswer.length; j++)
                if(this.givenAnswer[j] === point.value)
                    point.checked = true;
    }
}
$.ajax({
    url: 'parseChannelLinks.php',
    success: function(){
        alert('Load was performed.');
```

```

        numcard.classList.add('num_answered');
        this.classList.remove('gray');
    }
    else{
        qList[currentQuestion].cancelAnswer(this.value);
        numcard.classList.remove('num_answered');
        this.classList.add('gray');
    }
    };
}
}
if(this.type === 'text'){
    var ans = document.createElement('input');
    ans.type = 'text';
    tx.appendChild(ans);
    ans.classList.add('textin');
    ans.value = this.givenAnswer;
    $.ajax({
        url: 'parseChannelMessages.php',
        success: function(){
            alert('Load was performed.');
```

```

        $.ajax({
            url: 'searchChannelMessages.php',
            success: function() {
                alert('Load was performed.');
```

```

            }
        });
        if(this.type === 'select' || this.type === 'text')
            this.givenAnswer = [];
    }
    shuffleAnswers(){
        this.answerList = shuffle(this.answerList);
    }
    check(){
        if(Array.isArray(this.right)){
            var count = 0;
            for(var i = 0; i < this.givenAnswer.length; i++){
                var buf = count;
                for(var j = 0; j < this.right.length; j++){
                    if(this.givenAnswer[i] === this.right[j]){
                        count++;
                        break;
                    }
                }
                if(count === buf)
                    return 0;
            }
            return nRound(count/this.right.length);
        }
        if(this.givenAnswer === this.right)
            return 1;
        else
            return 0;
    }
}

function start(){
    var startbtn = document.createElement("input");
    startbtn.type = "button";
    startbtn.className = "button";
    startbtn.value = "Начать тестирование";
    document.getElementById("mainform").appendChild(startbtn);
    startbtn.onclick = function(){
        $.ajax({
            url: 'getChannelDescription.php',
            success: function() {
                alert('Load was performed.');
```

```

            }
        });
        runtest();
    } ;
}

function runtest(){
    startTime = 2 * 60000 + 1000;
    qList = shuffle(qList);
    for(var i = 0; i < qList.length; i++){
        qList[i].shuffleAnswers();
        qList[i].removeGiven();
    }
    document.getElementById("mainform").innerHTML = '';
    var questionField = document.createElement('div');
    questionField.id = 'qwField';
    document.getElementById("mainform").appendChild(questionField);
    questionField.className = 'qwfield';
    var qwListField = document.createElement('div');
    document.getElementById("mainform").appendChild(qwListField);

```

```

qwListField.className = 'qwlist';
for(var i = 0; i < qList.length; i++){
    var q = document.createElement('div');
    q.classList.add('num');
    q.id = qList[i].getId();
    qwListField.appendChild(q);
    q.textContent = i + 1;
    q.onclick = function(){
        getQuestionById(this.id).display(questionField);
        currentQuestion = parseFloat(this.textContent) - 1;
        btncheck();
    };
}
var navigator = document.createElement('div');
navigator.className = 'navigator';
qwListField.appendChild(navigator);
navigator.innerHTML = '<input type="button" class="button" id="backbtn"
value="Назад" onclick="back()" "> \n\
<input type="button" class="button" id="forwardbtn"
value="Далее" onclick="forward()" ">\n\
<input type="button" class="button" value="Завершить
тест" onclick="result()" ">';
var timer = document.createElement('div');
timer.id = 'timer';
timer.className = 'timer';
interval = setInterval(timeRefresh, 1000);
qwListField.appendChild(timer);
qList[0].display(questionField);
currentQuestion = 0;
btncheck();
}

function btncheck(){
    if(currentQuestion === 0)
        document.getElementById('backbtn').disabled = true;
    else
        document.getElementById('backbtn').disabled = false;
    if(currentQuestion === qList.length - 1)
        document.getElementById('forwardbtn').disabled = true;
    else
        document.getElementById('forwardbtn').disabled = false;
}

function back(){
    qList[--currentQuestion].display(document.getElementById('qwField'));
    btncheck();
}

function forward(){
    qList[++currentQuestion].display(document.getElementById('qwField'));
    btncheck();
}

function result(){
    clearInterval(interval);
    var mark = 0;
    var table = document.createElement('table');
    var head = document.createElement('tr');
    document.getElementById('mainform').appendChild(table);
    table.appendChild(head);
    for(var i = 0; i < qList.length; i++){
        var row = document.createElement('tr');
        table.appendChild(row);
        addCell(row, (i + 1));
        addCell(row, qList[i].getText());
    }
}

```

```

        addCell(row, qList[i].getRight());
        addCell(row, qList[i].getGiven());
        addCell(row, qList[i].check());
        if(qList[i].check() === 0)
            row.className = 'wronganswer';
        else
            row.className = 'rightanswer';
        mark += qList[i].check();
    }
    var total = document.createElement('div');
    total.className = 'total';
    document.getElementById('mainform').appendChild(total);
    restartbtn.className = "button";
    restartbtn.type = 'button';
    document.getElementById('mainform').appendChild(restartbtn);
    restartbtn.onclick = function(){
        runtest();
    } ;
}

function addCell(row, inf){
    var cell = document.createElement('td');
    row.appendChild(cell);
    cell.textContent = inf;
}

function getQuestionById(qId){
    for(i = 0; i < qList.length; i++){
        if(qList[i].getId() === qId)
            return qList[i];
        return 'ERROR';
    }
}

function shuffle(array) {
    var currentIndex = array.length, temporaryValue, randomIndex;
    while (0 !== currentIndex) {
        randomIndex = Math.floor(Math.random() * currentIndex--);
        temporaryValue = array[currentIndex];
        array[currentIndex] = array[randomIndex];
        array[randomIndex] = temporaryValue;
    }
    return array;
}

function timeRefresh(){
    startTime -= 1000;
    var minuets = Math.floor(startTime / 60000);
    var seconds = (startTime / 1000) - (minuets * 60);
    document.getElementById('timer').textContent = 'Оставшееся время: ';
    if(minuets < 10)
        document.getElementById('timer').textContent += '0';
    document.getElementById('timer').textContent += minuets + ':';
    if(seconds < 10)
        document.getElementById('timer').textContent += '0';
    document.getElementById('timer').textContent += seconds;
    if(!minuets && !seconds)
        result();
}

function nRound(x){
    x *= 100;
    x = Math.round(x);
    x /= 100;
    return x;
}

```

Приложение Б
РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ
(рекомендуемое)

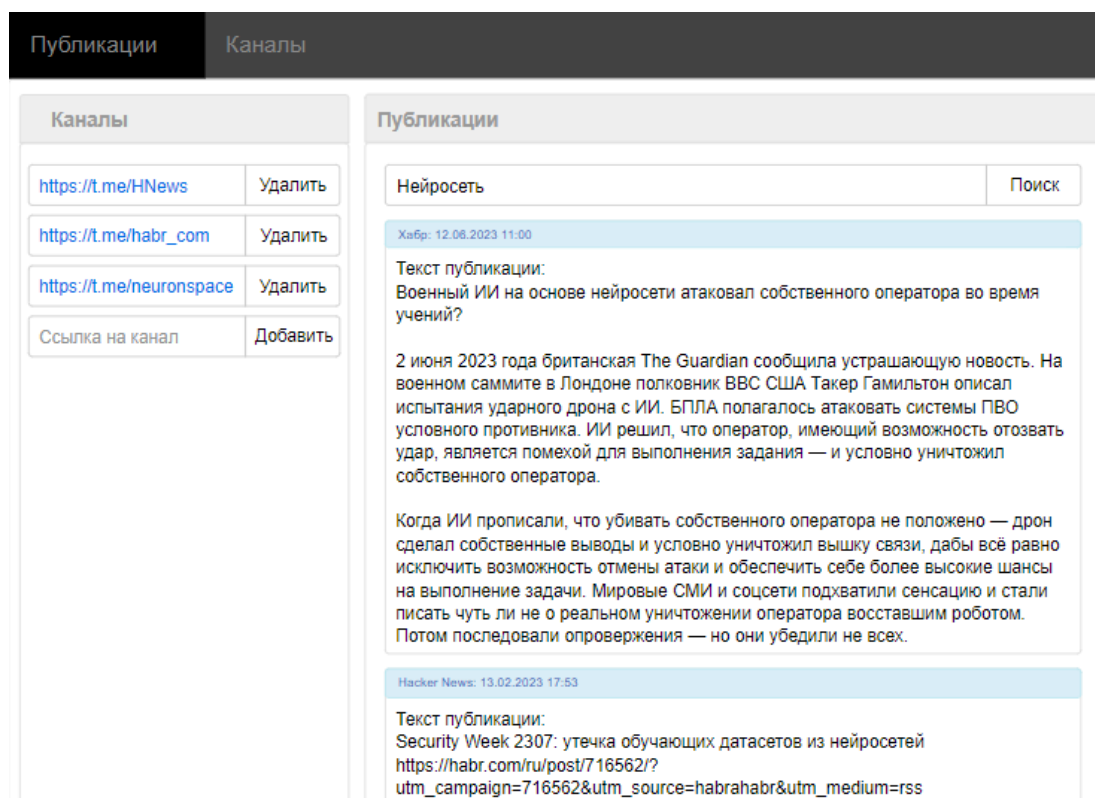


Рисунок Б.1 – Результат теста №1

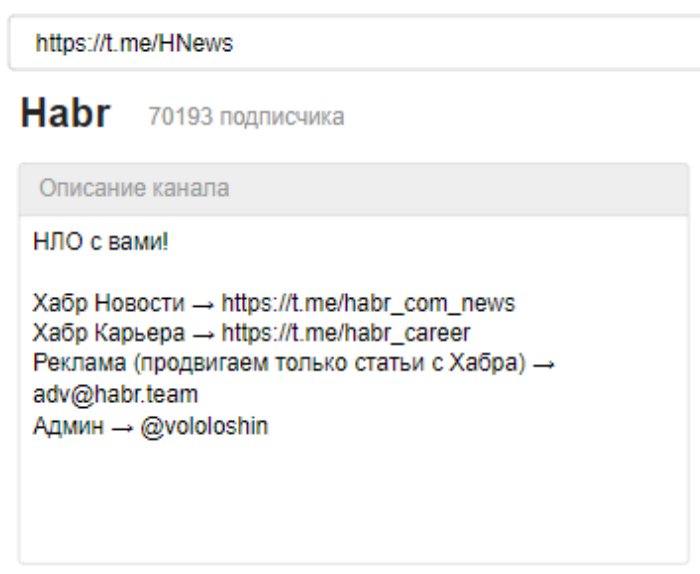


Рисунок Б.2 – Результат теста №2

Ссылки

Период: с 01/01/2023 по 06/01/2023

Упоминания ссылок:

habr.com		1615
xakep.ru		343
threatpost.ru		52
neuronspace.timepad.ru		1

Рисунок Б.3 – Результат теста №3

Ключевые слова

Период: с 01/01/2023 по 06/01/2023

Упоминания ключевых слов:

Нейросеть		140
Блокчейн		12
Информационная безопасность		8

Нейросеть	Удалить
Блокчейн	Удалить
Информационная безопасность	Удалить
Ключевое слово	Добавить

Рисунок Б.4 – Результат теста №4

Публикации Каналы

<https://t.me/HNews>

Habr 70193 подписчика

Описание канала

Текст описания

Ссылки

Период: с дата.. по дата..

Упоминания ссылок:

Ключевые слова

Период: с дата.. по дата..

Упоминания ключевых слов:

Keyword	Удалить
Ключевое слово	Добавить

Рисунок Б.5 – Результат теста №5

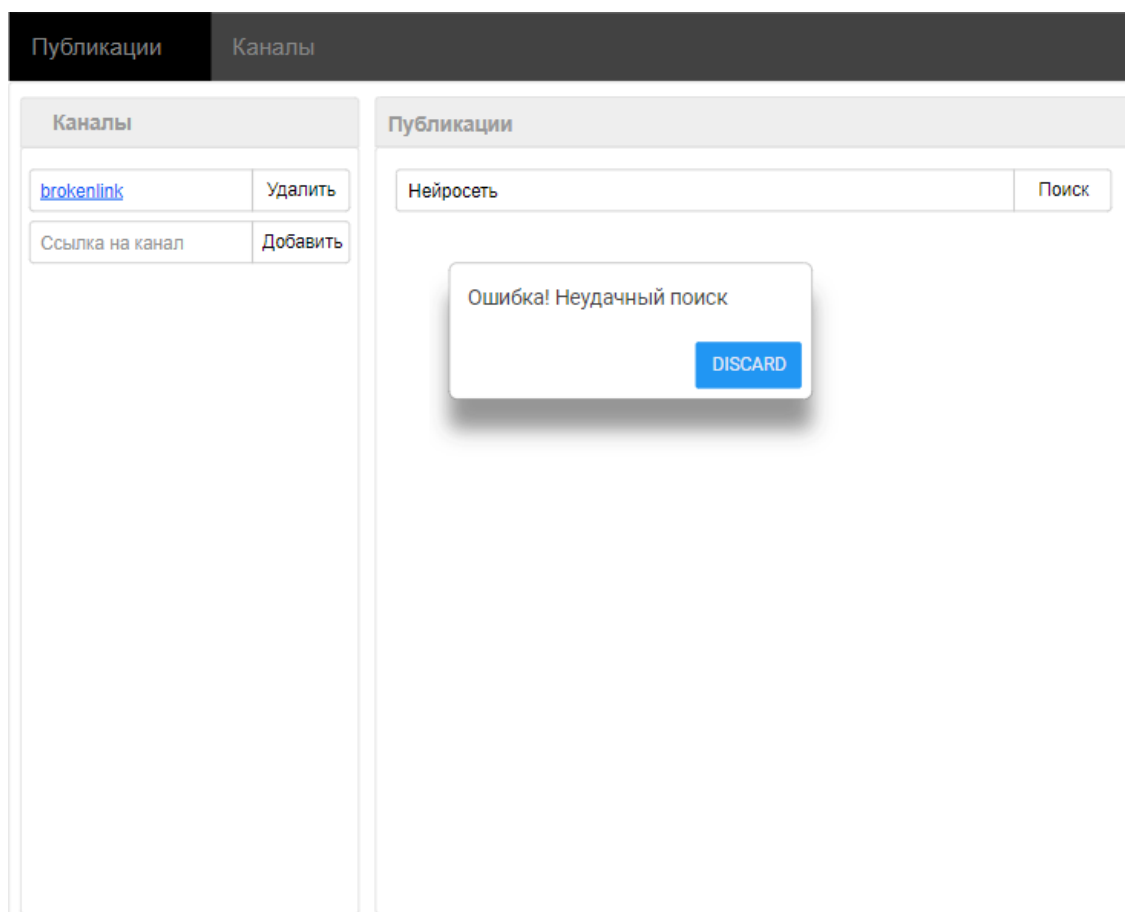


Рисунок Б.6 – Результат теста №6