

МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ
КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»
(СПбГУТ)

Факультет Инфокоммуникационных сетей и систем (ИКСС)

Кафедра Программной инженерии и вычислительной техники (ПИиВТ)

Допустить к защите

Заведующий кафедрой

(подпись) (ФИО)
« » 2024 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Разработка агрегатора социальных сетей для улучшения цифровой
коммуникации между пользователями

(тема ВКР)

Вид выпускной квалификационной работы дипломная работа
(дипломная работа, дипломный проект, магистерская диссертация)

Направление/специальность подготовки
09.03.04 «Программная инженерия»
(код и наименование направления/специальности)

Направленность (профиль)
Разработка программного обеспечения инфокоммуникационных сетей и систем
(наименование)

Квалификация бакалавр
(наименование квалификации в соответствии с ФГОС ВО)

Студент:
Саганенко Артемий Вадимович,
ИКПИ-04
(Ф.И.О., № группы) (подпись)

Руководитель ВКР:
Кандидат технических наук,
доцент кафедры ПИиВТ,
Белая Татьяна Иоанновна
(учёная степень, учёное звание, Ф.И.О.) (подпись)

(дата)

(подпись)

(ФИО студента)

Текст ВКР размещен в электронно-библиотечной системе университета.

Руководитель отдела комплектования библиотеки _____

(Ф.И.О.)

(дата)

(подпись)

Коэффициент оригинальности ВКР 95 % .

Проверил: _____

(Должность, Ф.И.О.)

(дата)

(подпись)

**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ
КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»
(СПбГУТ)**

Факультет Инфокоммуникационных сетей и систем (ИКСС)

Кафедра Программной инженерии и вычислительной техники (ПИиВТ)

Направление/специальность подготовки 09.03.04 «Программная инженерия»
(код и наименование)

Утверждаю:

Заведующий кафедрой

(подпись) (ФИО)
« ____ » _____ 20 ____ г.

ЗАДАНИЕ

на выполнение выпускной квалификационной работы (ВКР)

1. Студент Саганенко Артемий Вадимович № группы ИКПИ-04
(фамилия, имя, отчество)
2. Руководитель Белая Татьяна Иоанновна, доцент кафедры ПИиВТ, к.т.н.
(фамилия, имя, отчество, должность, уч. степень и звание)
3. Квалификация бакалавр
(наименование в соответствии с ФГОС ВО)
4. Вид ВКР дипломная работа
(дипломная работа, дипломный проект, магистерская диссертация)
5. Тема ВКР Разработка агрегатора социальных сетей для улучшения
цифровой коммуникации между пользователями
- утверждена приказом ректора университета от « 24 » апреля 2024г. № 534/к
6. Исходные данные (технические требования): C#, .NET, Visual Studio, API, OOP, LINQ,
desktop dev, социальные сети, цифровые коммуникации
7. Содержание работы (анализ состояния проблемы, проведение исследований, разработка,
расчеты параметров, экономическое обоснование и др.)
 1. Введение
 2. Анализ
 3. Проектирование
 4. Разработка
 5. Заключение
 6. Список использованных источников
 7. Приложение

8. Вид отчетных материалов, представляемых в ГЭК (пояснительная записка, перечень, графического материала, отчет о НИР, технический проект, образцы и др.): _____
пояснительная записка, презентация, приложение, электронный носитель

9. Консультанты по ВКР с указанием относящихся к ним разделов

Раздел	Консультант	Подпись дата	
		Задание выдал	Задание принял
Введение	Белая Т.И.	03.05.2024	07.05.2024
Анализ	Белая Т.И.	08.05.2024	11.05.2024
Проектирование	Белая Т.И.	12.05.2024	16.05.2024
Разработка	Белая Т.И.	17.05.2024	20.05.2024
Заключение	Белая Т.И.	21.05.2024	22.05.2024

Дата выдачи задания «03» мая 2024 г.

Дата представления ВКР к защите «22» мая 2024 г.

Руководитель ВКР _____
(подпись)

Студент _____
(подпись)

КАЛЕНДАРНЫЙ ПЛАН

№ п/п	Наименование этапов выпускной квалификационной работы (ВКР)	Срок выполнения этапов ВКР	Примечание
1	Постановка цели выполнения ВКР и задач	03.05.2024	выполнено
2	Работа с теоретическим материалом	04.05.2024 - 09.05.2024	выполнено
3	Сбор информации, необходимой для написания работы	10.05.2024 - 13.05.2024	выполнено
4	Систематизация и обработка материалов ВКР	14.05.2024 - 19.05.2024	выполнено
5	Анализ полученных в работе результатов, обобщение	20.05.2024 - 22.05.2024	выполнено
6	Подготовка отчетных материалов, представляемых в государственную экзаменационную комиссию, доклада к защите и презентации	23.05.2024 - 31.05.2024	выполнено
7	Консультации с руководителем ВКР	01.06.2024	выполнено
8	Представление выполненной ВКР руководителю для подготовки отзыва	02.06.2024 - 11.06.2024	выполнено
9	Подготовка к защите ВКР, включая подготовку к процедуре защиты и процедуру защиты	12.06.2024 - 17.06.2024	выполнено

Руководитель ВКР _____
(подпись)

Студент _____
(подпись)

РЕФЕРАТ

Дипломная работа содержит:

- 104 страницы;
- 2 таблицы;
- 10 рисунков;
- 7 приложений.

Ключевые слова: социальные сети, цифровые коммуникации, .NET, C#, LINQ, WTelegram, TDLib, Telegram API, VkNet, VK API, агрегация социальных сетей, интеграция социальной сети, разработка приложений десктоп, кроссплатформа, приложение для Windows, консольное приложение для Linux Ubuntu, работа с API, трансформация данных, ООП;

В ходе выполнения дипломной работы:

- Разработано программное обеспечение, позволяющее агрегировать сообщения из разных социальных сетей для улучшения цифровой коммуникации между пользователями;
 - Проведен анализ разработки социологического программного обеспечения в рамках цифровых коммуникаций, социальных сетей и в социальной сфере;
 - Выполнено полноценное проектирование разрабатываемой системы, с своевременным аргументированием, обоснованием выбранных технологий, методологий, алгоритмов;
 - Написана документация к разработанному ПО, описывающая способы взаимодействия человека и системы.
-

СОДЕРЖАНИЕ

СОДЕРЖАНИЕ	7
ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ТЕРМИНОВ	8
ВВЕДЕНИЕ	15
1 АНАЛИЗ	19
1.1 Актуальность разрабатываемого решения	19
1.2 Глобальная цель	23
1.3 Гипотеза	29
1.4 Цель	30
1.5 Задачи	33
1.6 Аналогии	37
1.7 Анализ области и объекта применения	41
1.8 Анализ разрабатываемой системы	51
2 ПРОЕКТИРОВАНИЕ	54
2.1 Проектирование решения	54
2.2 Используемые методологии и алгоритмы	64
2.3 Требования к системе	66
3 РАЗРАБОТКА	69
3.1 Общие сведения системы	69
3.2 Стек разработки и используемые технологии	70
3.3 Этапы разработки	72
3.4 Работа программы и интерфейс	76
3.5 Перспективы развития и нереализованный функционал	80
3.6 Недостатки системы	82
3.7 Руководство пользователя	84
ЗАКЛЮЧЕНИЕ	87
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	88
Приложение А	91
Приложение Б	92
Приложение В	93
Приложение Г	94
Приложение Д	95
Приложение Е	96
Приложение Ж	101

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ТЕРМИНОВ

API	Application Programming Interface. Прослойка в любой веб- или, реже, офлайн-системе, которая включает функции и их документацию для управления данными системы со стороны, по сути, описание способов использования и взаимодействия
Big data	Социально-экономический феномен, тесно связанный с data science. Понятие о размерности и сложности сбора и анализа данных, объясняющий набор алгоритмов для работы с их большим количеством
CIL	Common Intermediate Language. Промежуточный язык, используемый в .NET Framework и .NET Core. Программы, написанные на любом языке .NET: C#, VB.NET или F#, сначала компилируются в CIL, а затем выполняются на платформе .NET с помощью JIT-компиляции
Data science	Раздел информатики, заключающий в себе набор методов для работы с данными. Раздел, объясняющий и аргументирующий методы анализа данных, принцип применения данных, конечный выбор алгоритма на основании имеющихся
DBM	DataBase Manager. То же, что и СУБД

DRY	Do not Repeat Yourself. «Не повторяйся» — принцип разработки и написания кода, который подразумевает, что каждая часть кода должна иметь единственное, непротиворечивое и авторитетное представление в рамках целой системы
ERD	Entity Relationship Diagram. Диаграмма для отображения взаимодействия разных полей, элементов таблиц и других аспектов внутри БД
F#	Мультипарадигмальный, в первую очередь функциональный ЯП. В CIL компилируется наряду с C#, VB.NET в объектный файл.
FAQ	ЧАВО. ЧЗВ. Часто задаваемые вопросы. Перечень статично и преждевременно отвеченных вопросов по продукту или услуге для незамедлительного ответа в случае вопроса со стороны клиента
FIFO	First In, First Out. Первый попавший в структуру данных элемент первым из нее выходит
GUI	Graphical User Interface. Тип интерфейса пользователя, который позволяет взаимодействовать с аспектами ПО с помощью графических элементов, таких как окна, иконки, кнопки и меню, вместо текстовых команд
IDE	Integrated Development Environment. Интегрированная среда разработки. Набор инструментария для проектирования, разработки, развертки и внедрения ПО

JIT	Just-In-Time. Метод выполнения программ, при котором исходный код или промежуточный код (CIL в .NET) компилируется в машинный код непосредственно только перед его выполнением
KISS	Keep It Simple, Stupid. «Делай проще, тупица» — принцип написания кода, который подразумевает, что большинство систем работают лучше всего, если они остаются простыми, а не усложняются
LIFO	Last In, First Out. Принцип работы структуры данных — кучи. Последний элемент, который попал в очередь, первый из нее выходит. Один из самых часто используемых концептов структур данных. Прямая противоположность FIFO
LINQ	Language Integrated Query. Компонент платформы .NET, который добавляет возможность запросов к данным. LINQ позволяет писать тип-безопасные, читаемые и легко поддерживаемые запросы к различным источникам данных прямо в коде
MVP	Minimal Viable Product. Минимально жизнеспособный продукт. Самая первая рабочая версия продукта, решающая проблематику в очень ограниченном виде
PoC	Proof of Concept. Доказательство идеи. Конечная цель разработки той или иной системы, заключающаяся в доказательстве работы той или иной задумки. По своей сути MVP с исследовательским (эмпирическим, теоретическим) уклоном

PR	Public Relations. Связи с общественностью. Отдел каждой современной компании, нацеленный на работу с обществом, с рекламой, продвижением и поддержкой продукта
QoL	Quality of Life. Качество жизни — характеристика удобства и простоты пользования тем или иным ПО, отображающая, насколько нравится и удовлетворяет пользователя то или иное ПО
SMM	Social Media Marketing. Реклама и продвижение продукта через социальные сети. Ключевой элемент приобретения аудитории и продажи продукта в современном мире
SOLID	Single responsibility, Open–closed, Liskov substitution, Interface segregation, Dependency inversion. Расширенная трактовка парадигмы ООП, заключающаяся в следование указанных принципов при работе с классами, интерфейсами, объектами, абстракциями, имплементациями
Token	В аутентификации специальное поле, значение, обычно шестнадцатеричное число, которое отображает ключ для взаимодействия с полем внутри авторизованной области системы
UML	Unified Modeling Language. Язык графического описания для объектного моделирования системы в проектировании и/или разработке

UX/UI	User eXperience / User Interface. Два взаимосвязанных, но отдельных элемента GUI. UX — подразумевает опыт пользователя при работе с GUI, то есть эмоции, удобность, сам опыт использования. UI — подразумевает техническую, дизайнерскую реализацию
VCS	Version Control System. Система контроля версиями для отслеживания исходных файлов, используемых в ПО ресурсов, а также для удаленного/локального сохранения прогресса
WPF	Windows Presentation Foundation. Графическая подсистема для создания настольных приложений на платформе Windows. Реализует отображение, проектирование элементов интерфейсов в окне с помощью XAML
XAML	eXtensible Application Markup Language. Декларативный язык разметки, используемый для инициализации структурированных значений и объектов
YAGNI	You Aren't Gonna Need It. «Вам это не понадобится» — принцип написания кода, для предотвращения реализации ненужного функционала
БД	База Данных
Бот	Программа для автоматического выполнения указаний по заданному правилу и расписанию

Дайджест	Digest — усваивать. Краткий смысл того или иного материала или темы. Переименование с сильным сокращением для уменьшения времени восприятия и сложности понимания. Один из самых популярных видов изложения материала
Конструкт	Понятие из области социологии и философии, которое означает представление или идею, созданную обществом и признаваемую им как факт, хотя на самом деле основанную на соглашениях или договоренностях
Концепт	Идея
ООП	Объектно-Ориентированное Программирование. Парадигма программирования, основанная на концепции объектов
ОС	Операционная Система
Парсинг	Parsing — перебор с разбором. То же, что и семантический анализ.
ПО	Программное Обеспечение
Семантический анализ	Процесс анализа предоставленных данных, их категоризация, структуризация, нормализация для их дальнейшего использования
Соцсеть	Социальная сеть
СУБД	Система Управления Базами Данных

Технологическая сингулярность	Этап научно-технического развития человечества, когда контроль над процессами и элементами исследуемой и разрабатываемой системы перестает быть возможным по ряду научных, технических и когнитивных причин. В таком случае наблюдается экспоненциальный рост в развитии технологии. Точка Омега технологической сингулярности — это момент ее начала
Точка Омега	Философско-экзистенциальное понятие о моменте времени в развитии чего-либо, когда контроль и полноценный анализ процессов и переменных становится невозможным в свете всеохватывающей комплексности и стохастического самостоятельного инертного развития
Трансцендентность	Простым языком, что-то за гранью понимания, невозможное для понимания обычному человеческому уму
Трекинг	Tracking — слежение. Слежение и получение информации из данных, представленных на социальных платформах, имеет негативный окрас, выполняется в злых намерениях

Эффект Бабочки Понятие из теории хаоса, которое описывает явление, при котором малейшие изменения в начальных условиях динамической системы могут привести к далеко идущим и непредсказуемым результатам в будущем. Наличие у процесса данного эффекта указывает на некоторую стохастичность и/или неконтролируемость работы с системой

ЯП Язык Программирования

ВВЕДЕНИЕ

Полное наименование выпускной квалификационной работы: проектирование агрегатора социальных сетей и разработка приложения обобщенной системы для обмена сообщениями между разными пользователями различных социальных сетей для решения социологической проблематики в области цифровых коммуникаций и улучшения цифрового общения.

Система позволяет пользователю взаимодействовать с разными социальными сетями и использующими их людьми, используя один узел, данную систему. Пользователи могут получать и просматривать сообщения из разных источников, социальных сетей, от разных пользователей — клиентов социальных сетей, в условиях одного приложения (системы). Пользователи могут составлять и посылать сообщения с одной платформы любому другому пользователю через любую другую социальную сеть. Пользователи могут подключать, активировать агрегацию разных доступных ему социальных сетей. Пользователи имеют возможность персонализировать опыт, настраивать приложения под свои требования.

В целом потенциально пользователь может полностью заменить использование социальных сетей с помощью данной платформы, при этом сокращая время проведения в подобных сетях и улучшая опыт цифровой коммуникации, упрощая общение с людьми вокруг, если не в рамках разработанного по итогу работы приложения, то в рамках спроектированного концептуального решения.

Приложение направлено на экономию времени и прочих ресурсов, на упрощение социальных взаимодействий между людьми одного или разных слоев населения в условиях цифровых коммуникаций с помощью социальных сетей. Приложение потенциально решает определенный набор проблем. Это включает в себя решение разнообразных частных и глобальных проблем

различной важности. В том числе предлагает решение указанных далее проблем:

1. Глобальная проблема зависимости от социальных сетей;
2. Глобальная проблема обособленности цифровых сообществ;
3. Частная проблема затраты большего количества времени и ресурсов;
4. Частная проблема социализации определенного человека;
5. Частная проблема технологической эргономики и удобства пользования;
6. Дополнительная проблематика, связанная с объектом применения, сферой и рынком.

Глобальная цель — снизить времяпрепровождение людей в социальных сетях на 10%, увеличить количество активных пользователей на 10%, уменьшить количество неактивных контактов на 20%, увеличить количество друзей в социальных сетях на 10%.

Указанная дополнительная проблематика и частная проблематика технологической эргономики имеет специфичный вид и будет оговорена на разных этапах проектирования, а также в разделе недостатков. В особенности это касается дополнительной проблематики, в частности раздела с рынком.

Востребованность и актуальность тесно связаны с проблематикой, так как при условии решения частных проблем, проблем каждого человека по отдельности, среднему пользователю этот инструмент будет нужен, и он будет обладать достаточной долей релевантности, так как будет предлагать решения конкретным проблемам пользователя, обычного человека, помимо глобальных проблем, интересующих человечество.

Сфера применения технологии — социальная, объект применения технологии — цифровые коммуникации и социальные сети. Теоретическая значимость связана с анализом сферы и объекта применения, а также с гипотезой и обоснованием необходимости разрабатываемой системы. В ходе этапов теории решения и реализации необходимо аргументировать список выбранных технологий, методов и алгоритмов.

Система будет разрабатываться как десктопное приложение с визуальным интерфейсом, требующее постоянное подключение к интернету. Практическая значимость связана с разработкой работающей системы для агрегации контента социальных сетей, которая достигает поставленной цели и полностью или частично решает всю проблематику на практике.

Таким образом, работа имеет прикладной характер, где обоснование к разработке объясняется в ходе теоретического анализа сферы и объекта применения и выдвигается предположение о релевантности, востребованности и актуальности. После этого смысл заключается в правильной реализации спроектированной системы.

В ходе разработки и проектирования необходимо выполнять своевременно документацию технологии и методологии. В конце необходимо создать инструкцию пользования, выдвинуть системные требования к программному обеспечению. Разработка и проектирование должно также выполняться согласно в первую очередь календарному плану, а также этапам задач и этапам разработки.

Разрабатываемая система имеет аналоги, где каждый из них выполняет схожий функционал, однако все аналоги нацелены на другую аудиторию и, следовательно, функционал, его список и реализация несколько разнятся с наиболее эффективным и предпочтительным вариантом для клиентской базы разрабатываемой системы.

В ходе работы будет проведен анализ цифровых коммуникаций и социума для обоснования теоретической значимости разрабатываемой системы. В ходе работы будет выполнена разработка реализации приложения, выполняющего спланированный и спроектированный функционал.

Система в разработке, располагает определенными перспективами развития, имеется потенциал для улучшения независимого функционала системы и функционала агрегации и интегрирования с социальными сетями, также может быть расширена доступность и поддержка разного аппаратного обеспечения. Помимо этого, есть возможность дальнейшего улучшения

безопасности и надежности системы. Перспективы оговорены в соответствующем разделе.

В конечном итоге работа подразумевает создание продукта для определенной аудитории с доказанной релевантностью, актуальностью и востребованностью. Продукт не должен полностью решать проблематику во всей ее комплексности, масштабе и многогранности. Решение определенной проблемы из всей проблематики в определенной мере является достаточным фактором достижения цели проектирования и разработки.

Стоит также отметить, что разработка подразумевает создание MVP — Minimal Viable Product, не полностью функционального решения, которое в то же время должно отражать большую часть ключевого функционала. Или с другого ракурса — создания PoC — Proof of Concept, минимального работающего решения, которое в определенной мере доказывает гипотезу.

Необходимо отметить то, что теоретическая часть в контексте сферы социологии, и в аспекте цифровых коммуникаций и социальных сетей подразумевает анализ потенциальной разработки и поверхностный анализ процессов и феноменов сферы. В свете специфики специальности полноценный анализ социологических процессов будет некомпетентным и не будет иметь семантического веса в контексте работы. По этой же причине вектор анализа сдвигается на анализ именно процесса разработки в сфере и объекта применения системы.

Работа будет иметь соответствующие разделы, которые будут отвечать за разные аспекты выполненной работы. Три главные раздела: анализ, целиком теоретический раздел; проектирование, раздел, который включает в себе и прикладные, практические элементы, и теоретические; разработка включает в себя исключительно практическую часть работы.

1 АНАЛИЗ

1.1 Актуальность разрабатываемого решения

Разработка данной системы связана с актуальностью решения следующих глобальных проблем человечества: цифровой коммуникации между людьми, использующими разные социальные сети, которые имеют разные интересы и являются частями отдельных социальных групп; большого количества времени, используемого на получение, поиск и обработку информации, взятой из разных социальных сетей. Также в подобной степени с решением следующих проблем частного характера: потери личного времени при использовании разнообразных платформ социальных сетей; затраты финансовых и прочих ресурсов; сложности цифровой коммуникации в целом. В последней степени с решением дополнительной проблематики рынка. Частичное нивелирование данных проблем или же их полное устранение имеет свой смысл и достаточную важность. Следовательно, такая система располагает необходимым уровнем актуальности.

Проектируемая система является востребованной технологией в настоящее время и будет также востребованной в ближайшем обозримом будущем. Сегодня многие люди активно участвуют в социальных сетевых платформах и взаимодействуют друг с другом, используя подобные платформы постоянно, повседневно и исчерпывающе. При этом едва ли можно сказать, что социальные сети перестанут быть всеобъемлющими и популярными в ближайшее время. Технология социальных сетей, можно с уверенностью заявить, стала ключевой для функционирования общества 21-го века. При этом в своем корне технология неотделима от социологических и коммуникативных процессов. Сложно представить современное общество без социальных сетей, а современное общение — без цифрового. По этой же причине любое улучшение социальных платформ при должном исполнении будет иметь свою релевантность и аудиторию. Такое улучшение также может быть

монетизировано. Это означает, что объект применения, то есть социальные сети и цифровые коммуникации, актуален и востребован.

Для улучшения подобной цифровой коммуникации, то есть текстовой и файловой коммуникации в контексте одной или нескольких платформ, могут быть предложены и впоследствии реализованы дополнительные методологии и технологии. В свете глобальности социальных сетей и того факта, что они затрагивают все аспекты жизни как общества в целом, так и каждого человека в частности, проектирование и интеграция подобных методологий могут повлиять как на цифровые коммуникации, так и на само общество и его внутренние процессы. Например, могут создавать дополнительные точки конфронтации между разными субкультурами, производить внедрение новых позиций для социальных взаимодействий, сказываться на качестве и скорости общения, воздействовать на скорость ассимиляции и прочих социальных процессов, в целом оказывать влияние на социум. Это в конечном итоге означает, что при реализации потенциальных улучшений в данном контексте можно говорить об изменениях в цифровых коммуникациях, что в свою очередь, скорее всего, приведет к небольшим отклонениям в социологических процессах. При положительном характере изменений можно выдвинуть предположение об усовершенствовании цифровых коммуникаций и об изменении некоторых отдельных элементов социума. Например, можно выдвинуть предположение об уменьшении барьера стереотипов, ускорении общения во временном характере. Однако для достижения идеи повышения скорости и гибкости цифровых коммуникаций следует принимать во внимание не только особенности каждой социальной сети как отдельной платформы, но и особенности взаимодействия людей в обществе. Необходимо иметь в виду все виды коммуникаций и учитывать социум во всей присущей ему многогранности, неоднозначности, глобальности и неизмеримости. Только в случае, если все особенности и виды коммуникации учтены, можно говорить о преднамеренном улучшении цифровых коммуникаций.

Помимо указанных ранее преимуществ в виде экономии времени, затрат и упрощения общения, данная технология предложит широкий спектр новых решений для достижения общей цели улучшения опыта общения между лицами независимо от контекста: финансовой, временной, географической и социальной ситуации. Таким образом, такая технология также уменьшит технологический, финансовый и классовый разрыв населения, предоставляя более гибкие условия для общения, достигая при этом цели объединения людей, предоставляя им возможность общаться. Идея заключена в предоставлении технологии для людей с разными интересами и социальными положениями общаться друг с другом, игнорируя навязанные барьеры платформ социальных сетей. Таким образом, система обладает необходимой релевантностью.

Агрегатор может стать новым решением, которое может применяться обычными людьми для повседневного рутинного общения и для сокращения затрат времени на проверку новой информации на платформах. Обычные пользователи социальных платформ являются главной клиентской базой для данной системы. Такая система может также использоваться для SMM, трекинга, семантического анализа, организации аккаунтов знаменитостей, ведения аккаунтов для продукции, реализации ботов. При этом упомянутые способы использования не являются приоритетными, что значит, что они не будут поддерживаться напрямую, а лишь косвенно продолжать работать. Такие функциональные дополнения являются следствием особенностей технологий каждой из платформ социальных сетей и самой разрабатываемой технологии агрегатора. В целом, можно считать, что эти варианты использования являются отдельными, незначительными дополнениями к актуальности и релевантности разрабатываемой системы. Заклучая, система обладает необходимой востребованностью среди людей.

Значимость практической составляющей заключена в разработке системы с функционалом, который способен решать глобальные и частные проблемы, поставленные в разделе целей проекта. В то же время

теоретические элементы внедрены в решение указанных практических на локальном уровне. Такие локальные проблемы решаются повсеместно в ходе проектирования и разработки и будут указаны в каждом отдельном пункте разработки. Стоит вновь упомянуть о проблемах социального и гуманитарного характера, для решения которых данная система разрабатывается. Социальный и гуманитарный аспект тоже имеет скорее теоретический характер, так как эмпирическое, прикладное исследование эффективности не может быть проведено в свете особенностей самой сферы и специальности. Теоретический и практический аспекты имеют достаточный, соответственно, исследовательский и прикладной вес.

Для уточнения, чтобы избежания недопонимания и для логичности текста, обоснования и аргументации, дадим следующие определения. Актуальность — характеристика, отражающая современность и своевременность разрабатываемой системы, она отражает, насколько в конечном итоге система нужна на сегодняшний день. Востребованность — характеристика, отражающая необходимость такой системы, насколько человеку, миру она необходима. Релевантность — характеристика, которая отображает степень соответствия желаемого и действительного в контексте программного обеспечения. Все три параметра хоть и заключают разные аспекты, тем не менее в определенной мере взаимозаменяемы и могут использоваться как синонимы в данной работе.

Подводя итог, система агрегатора платформ социальных сетей обладает актуальностью, релевантностью и востребованностью. Система располагает набором частных проблем, решение которых интересует каждого человека по отдельности. Система располагает набором глобальных проблем, в решении которых заинтересовано общество в целом. Разрабатываемая методология, технология реализуется в крайне популярной сегодня и обозримом будущем сфере — цифровых коммуникаций, также агрегатор является надстройкой для социальных сетей с большим охватом аудитории и популярностью. Система проектируется на базе уже задействованных готовых технологий и может быть

далее расширена. Целевая аудитория системы — это обычные повседневные пользователи социальных сетей. Проектируемая система имеет практическую и теоретическую значимость. Следовательно, система должна обладать следующими качествами: модульность, по причине надстройки над другими социальными платформами; эффективность, по причине разработки системы цифровых коммуникаций; защищенность по причине разработки, работающей с пользовательскими данными; глобальность по причине работы с социальными сетями.

1.2 Глобальная цель

Разработка данной системы и реализация функционала для достижения конечной цели решает как частные проблемы, так и глобальные. Глобальные проблемы или же глобальная цель — это комплекс, совокупность проблематики гуманитарного, социального, глобального и планетарного характера, затрагивающая интересы всего человечества, от решения которых зависит прогресс или регресс, сохранение цивилизации. Глобальные проблемы взаимосвязаны и имеют высокий динамизм, гибкость, частичную стохастичность, затрагивают все сферы жизни, затрагивают бóльшую часть населения. Подобные проблемы исчезают и появляются в зависимости от экологической, технической и социальной ситуации в мире.

Наличие глобальной цели характеризует разрабатываемый проект как всеобъемлющий и релевантный, а также отражает его общую концепцию. Глобальная цель определяет вектор задач и общий смысл разработки конкретного ПО — программного обеспечения. Если система частично или полностью решает одну из первоначально обозначенных глобальных проблем, тогда система как минимум актуальна. Подобная система при качественной реализации гарантировано найдет свою аудиторию, спрос и, при условии наличия монетизации, финансовый доход.

Проектируемая и разрабатываемая система реализуется в сфере цифровых коммуникаций и будет взаимодействовать с аспектами общества,

социальными процессами, обменом информации и данных и коммуникациями в обществе. Следовательно, затрагиваемые глобальные проблемы также по большей части будут социальными.

Как уже и было сказано, решение глобальных проблем или наличие глобальных целей — необходимый фактор для того, чтобы обозначить проектируемую систему как актуальную и релевантную. Однако не стоит выпускать из виду частные проблемы, которые среди людей имеют больший приоритет. В таком случае, если глобальные и частные цели будут поставлены правильно, а также будут предприняты попытки достижения всех этих целей, тогда проект будет востребованным. Для достижения этого выведем глобальные проблемы, которые данная система может и будет решать.

Одной из глобальных целей проектируемой технологии является уменьшение времени использования социальных сетей. Согласно множеству исследований, связанных с социальными сетями, человечество действительно имеет проблему всеобъемлющего и нерелевантного использования данных платформ. В этих исследованиях описываются нарушения социального строя, психического состояния, физического состояния, увеличение прокрастинации, тревоги, ухудшения состояния пользователей таких социальных сетей, разрушение принципов и институтов общества, чрезмерные затраты на электроэнергию, нагрузка на зрение, психику и так далее.

Согласно исследованиям издания «Mail.Ru Group» (сегодня «VK») и «comscore» от марта 2014 года об использовании социальных сетей в России^[13], среднее время пребывания пользователя «VK» на платформе составляет 10 минут, пользователя «Одноклассников» — 20 минут, неназванной иностранной социальной платформы — 8 минут. В сумме 38 минут в день.

В 2022 наблюдалась тенденция на ухудшении ситуации, и, согласно официальному пресс-релизу «VK» и «Mediascope»^[14], количество минут, проведенных в социальной сети «VK», поднялось до 20 минут, больше на 10 минут за 8 лет, либо до 46 минут, больше на 36 минут за 8 лет, среди

постоянных пользователей. В 2023 году ситуация только ухудшилась, и, по данным аналитического агентства «Mediascope»[\[31\]](#), средний пользователь интернета тратил на социальные сети уже около часа, что в среднем на 22 минуты больше, чем в 2014. Тут необходимо заметить, что нас больше интересует общее время, а не время, проведенное на определенной платформе. Например, к 2013 году был разработан «Телеграм», который начал выигрывать аудиторию у «ВК», нас данный факт не сильно интересует, так как отражает рынок, а не общую социальную картину.

В ходе коллегии Министерства цифрового развития, связи и массовых коммуникаций Российской Федерации от 24 апреля 2015 года был представлен доклад на тему социальных сетей и в целом рунета на территории РФ[\[22\]](#). Данный доклад характеризует развитие социального интернета в России. В докладе приведены цифры для отражения ситуации на рынке. Например, 74 миллиона человек заходили в интернет хотя бы раз, а 60 миллионов делают это каждый день. Исходя из этого, можно предположить, что в день население теряло от 60 миллионов до 74 миллионов человеко-часов полезной деятельности, работы из-за социальных сетей. Также в докладе упоминается процентное соотношение затраченного времени, таким образом, около четверти проведенного в интернете времени приходилось на социальные сети.

Исходя из тенденции, можно выдвинуть предположение, что сегодня, в 2024 году, время, затрачиваемое на интернет и социальные сети, еще больше, и сегодня человек тратит, используя их, так много, как никогда до этого в истории научно-технического и социального прогресса.

Подводя итог, можно выдвинуть предположение, что данная проблема действительно имеет скорее глобальный характер, нежели частный. Речь идет не просто о потере какого-то личного времени, а о нарушениях в социальных процессах из-за времени и ресурсов, затрачиваемых на использование социальных сетей. Судя по тенденции, время использования социальных сетей и интернета, скорее всего, не опустится до введения новой прорывной технологии коммуникации и социальной интеграции.

В статье [\[27\]](#) описывается социальное обособление, замещение реального мира и отождествление с реальными социальными процессами. В конечном итоге это приводит к стандартизации личности и ее приведению к общепринятому образу. А также деградации личности, с в первую очередь физиологической точки зрения. Данный конкретный пример не включает в себя временную часть использования сетей, но подразумевает, что общение в интернете быстрее, но в этой своей особенности негативно и приносит только вред как личности, так и обществу.

Список физиологических и психических проблем в [\[17\]](#) немного исчерпывающий и преувеличенный, в половине из указанных пунктов социальные сети имеют скорее косвенный или сопутствующий характер. Однако даже в таком случае, как фактор получения того или другого недуга, социальные сети следует переиначить в целях добиться не столько повышения эффективности цифрового общения, сколько даже сопутствующего уменьшения времени и упрощения самого общения.

Наивно предполагать о возможности перекройки всего общества и достижении полного игнорирования социальных платформ как феномена. В то же время не стоит оставлять без внимания отрицательные стороны технологии. В конечном итоге всё, что остается, — это улучшения того, что есть у людей на данный момент. В плане затраченного времени его сокращение любым методом является адекватной и полноценной задачей. Разрабатываемое ПО — программное обеспечение — имеет возможность решить эту глобальную проблему указанным способом. Данный функционал может быть разработан в ходе практической реализации главной задумки. Как минимум, объединив разнообразные источники информации, социальные платформы в единой системе, тем самым попытаться сократить время, проведенное в социальных сетях.

Главная проблема многих социальных сетей — это их кругозор, говоря более прямолинейно, — аудитория. Пользователи каждой из сетей — это

конкретный стереотип человека с конкретным набором требований, друзей, с определенной работой, возрастом, полом.

Самые массовые социальные сети не могут похвастаться охватом аудитории. «ВК», самая популярная в России социальная сеть, согласно указанной в дайджесте [\[4\]](#) статистики из исследования «Mediascope» от 2023 года, охватывает около 80 миллионов человек, что всего лишь 54% от населения России (2023 г.)[\[33\]](#), что является очень маленьким показателем. Имеется в виду, что этого значения недостаточно для определения «тотальная глобальность» и «тотальная массовость».

Компании не зря собирают статистику на тему пола, возраста, города и так далее, эти данные формируют эту цель — стереотип, там отчетливо видно, если сеть непопулярна среди старшего поколения или среди художников. Некоторые сети стремятся к максимальному охвату для максимальной прибыли, но едва ли могут это реализовать, так как не попавшие под стереотип цели остаются вне платформы. Другие сети, наоборот, стремятся занять определенную нишу и попытаться не допустить к себе конкурентов, но едва ли могут похвастаться размером аудитории и часто имеют проблемы с привлечением новых пользователей. Это является скорее нормой, и в этом есть свои плюсы и минусы. В то же время вряд ли с этим можно что-то поделать.

Малый охват создает среду для создания альтернатив: как нишевые проекты, так и массовые могут быть разработаны в таких конкурентных условиях. Но иметь несколько с большими аудиториями и хорошего качества определенно лучше, чем иметь сотни социальных сетей на пару сотен человек и плохого качества.

Такой излишне капиталистический подход на свободном рынке социальных сетей часто создает незаполненные места, для которых есть спрос, но нет предложения. Главная цель каждого конечного продукта и бизнеса — это заработок денег, это имеет отдельную силу на капиталистическом рынке. В таких условиях создавать продукцию для всех невозможно. Учитывая это, каждая соцсеть — социальная сеть, и ее разработчики нацелены на группу

людей, что приводит к тому, что социум нарочито делится на условные единицы по интересам с повышенным порогом вхождения, с дефицитом или профицитом аудитории. Далее такое деление приводит к социальному разрыву населения и множеству сопутствующих социальных отклонений, скорее с отрицательным окрасом, нежели чем с положительным. В таких реалиях общение между разными пластами населения очень затруднено и в ряде случаев невозможно. Пользователи неспособны общаться на многие темы по той причине, что узконаправленные специализированные сети не располагают аудиторией для поддержания общего любительского диалога.

Объясняя на примере, человек, заинтересованный в сообществе для художников и программистов одновременно, не имеет альтернативы, кроме как прыгать с одной соцсети к другой. В конечном итоге по разным причинам, без возможности или из-за малой эффективности коммуникаций в сообществе человек теряет свой интерес к какой-либо группе.

Наличие человека в соцсети — это определяющий фактор соцсети. Без него технология становится никому не нужной, а все вложенные ресурсы и средства становятся потерянными. Обратная ситуация: чем больше людей используют соцсеть, тем более она востребована в обществе, интересна человеку, имеет больше смысла[26]. Это очевидная особенность, но она закрепляет идею социальной сети: чем больше заинтересованных и активных людей используют соцсеть, тем больше в ней заинтересованных и активных людей.

Таким образом, создавая новые возможности для людей взаимодействовать друг с другом, упрощая, ускоряя процесс коммуникации, можно предположить о росте количества точек взаимодействия. Иначе говоря, двум случайным людям на земле становится проще найти друг друга, этот факт также изменяет общество, улучшая гибкость, свободу, коммуникабельность и упрощая сам процесс.

Заключая всё вышесказанное, проблематика, которую данная система предположительно в какой-то степени нивелирует, состоит из очень простых,

но важных элементов социума. Выводится две в максимальной степени приземленные проблемы: проблема чрезмерного использования социальных сетей и проблема рынка этих соцсетей. Решение проблем сводится к снижению чрезмерного использования соцсетей и к упрощению их использования клиентами вопреки рынку, соответственно. Таким образом, у разрабатываемой системы есть две глобальные проблемы, которые она должна решать.

1.3 Гипотеза

Разработка и внедрение программного обеспечения для агрегации контента социальных сетей позволит упростить и повысить эффективность цифровых коммуникаций. Это улучшит социализацию, сэкономит время и снизит зависимость от соцсетей.

Такое ПО станет инновационным решением для объединения различных сообществ. Оно также улучшит технологическую эргономику и может снизить затраты пользователей на использование социальных сетей. Благодаря этому программному обеспечению пользователи смогут легче получать доступ к информации из разных источников, что поддержит разнообразие, открытость и свободу в цифровых коммуникациях.

Помимо персональных преимуществ для каждого человека, разрабатываемая система также может оказать глобальное влияние на человечество. Она может улучшить гибкость рынка в современных реалиях.

Хотя сильные стороны системы могут показаться незначительными из-за её простоты, эффекты от её использования будут суммироваться и иметь кумулятивный характер. Это означает, что система может оказать значительное влияние, несмотря на свою идейную элементарность.

1.4 Цель

Целью данной работы является проектирование системы и последующая разработка программного обеспечения, нацеленного на агрегацию содержания социальных сетей пользователя. Система должна в первую очередь предоставлять возможность пользователю отправлять сообщения в разные источники из одного приложения и получать входящие сообщения из разных источников. При этом должны быть решены или частично решены следующие глобальные и частные проблемы:

- Глобальная проблема зависимости от социальных сетей и потери большого количества времени в ходе их использования;
- Глобальная проблема замкнутости сообществ и технологического, общественного и идейного обособления разных групп населения;
- Частная проблема затраты большого количества времени и, возможно, финансовых ресурсов при использовании социальных сетей;
- Частная проблема социализации и упрощения цифровых коммуникаций;
- Частная проблема технологической эргономики и простоты использования представленных технологий;
- Дополнительная проблематика связанная с переполнением свободного рынка, низким качеством предоставляемых услуг и чрезмерно высокой конкуренцией в сфере.

Стоит отметить, что решение каждой из указанных проблем может быть реализовано напрямую, то есть конкретным разработанным функционалом, либо же косвенно, то есть параллельно с функционалом. Специфика области применения сводит решения большинства проблем из указанных к косвенной реализации. При работе с коммуникациями и обществом и при разработке ПО для цифровых коммуникаций происходит разработка в конкретике социальных процессов. В данной работе не проектируется новый способ общаться, следовательно, и прямого социального функционала нет. Разрабатывается лишь метод улучшения и ускорения того, что есть на рынке в данный момент. Весь функционал интегрируется в ПО и/или ссылается на другое ПО.

Для достижения конечной цели разработки и проектирования необходимо привести доводы и аргументы к принятым на пути к цели решениям. Необходимо обосновать структуру, технологии, концепт, дизайн, теорию и документацию. В конце работы должны быть предоставлены: документация, код, реализация (сборка), инструкция пользователя, концептуальный документ, дизайн-документ, теоретическое обоснование, практическое обоснование, отчет, документ требований, документ системных требований, тезис (абстракт), титульный лист, календарный план, лист согласований сведений, реферат. Отсутствие какого-либо элемента из указанных дискредитирует в своей степени работу, снижая ее релевантность, надежность, эффективность.

Необходимо поэтапно и планомерно выполнять задачи, вывести заключение, добавить источники. В целом, данный проект должен разрабатываться согласно регламенту и согласно списку задач.

Практическая цель заключается в проектировании и разработке системы агрегатора. Теоретическая цель заключается в обосновании и выборе методов и технологий, а также в решении поставленных проблем. Проблемы имеют социальную специфику, измерение эффективности эмпирическим путем маловероятно.

Данный проект должен будет воздействовать на и работать с недостаточной распределённостью и сильной централизованностью социальных платформ и их аудиторий. Будет пытаться разрешать сопутствующие рынку сегодня недостатки. Также проект подразумевает теоретическую работу с социумом и рынком услуг. Иначе говоря, подразумевает решать или предпринять попытки решения следующих недостатков и изъянов платформ социальных сетей:

- Стереотипирование аудитории каждой из социальных сетей, ее приведение к общему стандарту, шаблону;

- Существование искусственных барьеров для общения между людьми, использующими разные платформы, обособление социальных групп, их сегрегация, разделение на основании личных интересов;
- Излишнее разнообразие и дикий капитализм на рынке в условиях рыночной экономики, переизбыток продуктов, отсутствие лимитов, отсутствие порога вхождения для каждой платформы;
- Нежелание издателей и разработчиков находить компромиссы в процессе проектирования платформ и создания концептов;
- Последствие свободного рынка и поколения потребителей в контексте поиска и использования социальной платформы, отсутствие приоритета эффективности и качества ПО, чрезмерный приоритет компаний на PR и маркетинге;
- Последствие главной идеи бизнеса — заработка денег, игнорирование глобальных проблем, нежелание бороться с изъянами, которые могут воздействовать на доход.

Таким образом, имея столь внушительный список изъянов и при условии исправления какой-то доли из них, концепт проекта идет немного против течения, против современных устоев корпораций и бизнеса. Имеет, грубо говоря, немного анархичный окрас. Тем не менее вряд ли имеет такой сильный эффект, что привлечет внимание платформ социальных сетей.

В общем и целом, разрабатываемая система создается для людей, общества и не в целях зарабатывать деньги, а в целях упрощения жизни и повышения эффективности отдыха, работы, морального и физического состояния.

Подводя итог, можно сказать, что данный проект имеет большой набор целей как прикладного, так и социального характера. Начиная с реализации в коде и дизайне, заканчивая цифровыми коммуникациями и социальными процессами. Этот набор исчерпывающ и содержит полную информацию о главной задумке и концепте проекта.

Цель любой работы — это решение конкретных проблем, указанных в списке решаемых. Проблемы, которые проектируемое ПО должно решать, были выдвинуты выше. Как только все проблемы будут либо решены, либо в достаточной степени исправлены, как только будет приведена документация решения, ее реализация и аргументация используемых методов, тогда и только тогда цель можно будет считать достигнутой.

1.5 Задачи

Задачей данной работы является проектирование и разработка системы агрегатора данных из разных социальных сетей, которая располагает функционалом для получения и отправления сообщений между пользователями разных социальных систем. В ходе проектирования и разработки системы необходимо использовать аргументированный набор инструментов, технологий, формул и алгоритмов. В конечном итоге, по факту выполнения всех задач, цели должны быть достигнуты, иначе либо задачи не целиком охватывают процесс решения проблем, либо набор целей/проблем имеет неверный характер.

Для достижения целей проекта необходимо выполнить все смежные, малые и главные задачи. Наличие четкого набора задач в начале проектирования и разработки повысит эффективность разработки и последующее качество ПО. В таком случае разработка будет быстрее, а финальный результат будет наиболее удовлетворяющим требованиям. Создание списка задач для цели происходит непосредственно перед началом проектирования проекта. Список должен быть исчерпывающим, но не преувеличенным. Задачи должны быть самоочевидными и покрывать 100% работы.

Проект следует выполнять поэтапно. Каждый из этих этапов целиком и полностью зависит как от факта выполнения предыдущего этапа, но, что не менее важно, от качества его выполнения. Таким образом, невозможно писать скрипты и строить взаимодействие элементов без разработанного алгоритма и

структуры. Таким образом, если алгоритм выбран правильный, то и скрипт будет работать эффективно. Этапы выполнения и задачи указаны далее.

1. Концепт

- 1.1. Определить сферу и объект применения;
- 1.2. Определить глобальные и частные проблемы системы;
- 1.3. Создать предположительный набор технологий, методологий и исследований, необходимых для решения проблемы;
- 1.4. Разработать концептуальный документ. Концептуальный документ поверхностно отражает смысл системы, кратко отвечая на вопросы: как, зачем, что, почему и подобные;
- 1.5. Доказать актуальность, релевантность и востребованность;
- 1.6. Выдвинуть гипотезу и теорию, поставить цель и задачи;

2. Проектирование

2.1. Анализ области применения

- 2.1.1. Провести анализ сферы и объекта применения;
- 2.1.2. Исследовать проблематику;
- 2.1.3. Провести анализ аналогов;
- 2.1.4. Определить принцип работы со сферой применения и провести анализ системы в разработке;

2.2. Теория решения

- 2.2.1. Уточнить принцип решения;
- 2.2.2. Аргументировать и объяснить интеграцию в объект исследования;
- 2.2.3. Исследовать эффективность, совместимость теории решения;
- 2.2.4. Проверить реализуемость;
- 2.2.5. Провести глубокий анализ решения. Обоснование и объяснение каждого из предыдущих этапов для конечного определения полного принципа решения;

- 2.2.6. Создать документацию решения. Полное описание спроектированного решения;

2.3. Теория реализации

- 2.3.1. Создать техническое решение, основанное на теории решения, на бумаге;
- 2.3.2. Выбрать методологии и алгоритмы реализации;
- 2.3.3. Исследовать эффективность реализации;
- 2.3.4. Исследовать технологии и создать стек разработки;
- 2.3.5. Создать документацию, которая будет описывать специфику принятых решений;

2.4. Теоретическое решение

- 2.4.1. Удостовериться, что у нас есть решение на бумаге, которое может исправить указанную проблематику. Это решение должно включать в себе отношения теории решения и теории реализации и объяснять весь принцип работы. В таком случае можно переходить к этапу разработки и реализации. Иначе необходимо еще раз пройти через все этапы проектирования;

3. Разработка

- 3.1. Настроить среду разработки. Создать решение;
- 3.2. Разработать каркас для ПО. (Back-end);
- 3.3. Разработать и протестировать ключевые функции. (Back-end).
Внутри каркаса необходимо разработать и проверить работу ключевых функций. Набор ключевых функций должен быть коротким, а их реализация безупречной. Также разработанный функционал должен обладать максимальной совместимостью и быть в максимальной степени абстрактным;
- 3.4. Разработать самые главные варианты использования. (Back-end).
Разработка методов и функций для Critical Path, для самого

утилизируемого пути использования программного обеспечения.

После завершения ПО должно выполнять свою главную задачу;

- 3.5. Создать базовый интерфейс ПО. (Front-end);
- 3.6. Реализовать дополнительный функционал. (Back-end). Достижение 100% покрытия функционала. Все функции к этому этапу должны как минимум быть реализованы, как максимум — работать;
- 3.7. Разработать интерфейс. (Front-end). Создание интерфейса в едином дизайне. Дизайн и создание интерфейса в том виде, в котором планируется предоставление ПО клиенту;
- 3.8. Улучшить технологическую эргономику. (Fullstack). Работа с интерфейсом и технической составляющей. Улучшение опыта использования. Достижение максимально удобного и красивого внешнего вида разрабатываемой системы и достижение максимальной пользы от ПО в контексте предметной области;
- 3.9. Выдвинуть набор системных требований;
- 3.10. Улучшить QoL — Quality of Life и техническую производительность. (Fullstack). Рефакторинг, оптимизация кода, устранение визуальных багов, добавление локализации и подобного функционала для улучшения опыта использования;
- 3.11. Выполнить документацию реализованного функционала, использованных методов, структур и технологий;

4. Тестирование

4.1. Тестирование сопутствующих систем методом черного ящика

- 4.1.1. Протестировать используемый Application Programming Interface функционал;
- 4.1.2. Провести интеграционное тестирование связей;

4.2. Тестирование главной системы методом белого ящика

- 4.2.1. Провести тестирование критического пути;
- 4.2.2. Выполнить смюк-тестирование ключевого функционала;
- 4.2.3. Провести интеграционное тестирование связей;

- 4.3. Выполнить нагрузочное тестирование. При условии работы с множественным доступом;
- 4.4. Выполнить тестирование безопасности. При условии работы с персональными данными;
- 4.5. Выполнить регрессионное тестирование. Необходимо при условиях обновления как используемого ПО, так и разрабатываемого;
- 4.6. Занести результаты в тест-кейсы;
- 4.7. Создать отметки о тестировании в документации;

5. Заключение

- 5.1. Завершить документацию;
- 5.2. Создать руководство пользователя;
- 5.3. Создать лицензию и/или патент;
- 5.4. Сделать вывод о проделанной работе. При условии удачи в разработке ПО и полной или частичной удачи в подтверждении гипотезы и цели разработки. Следует сделать вывод о работе, перечислив конкретно, что было достигнуто;
- 5.5. Создать список использованных источников;
- 5.6. Создать список использованного ПО;
- 5.7. Создать список авторских прав;

Пункты 5.5, 5.6, 5.7 должны быть выполнены согласно требуемым спецификациям, в том числе ГОСТ[\[8\]](#), Creative Commons[\[35\]](#), Harvard[\[37\]](#), DOI[\[36\]](#) и прочих. Все авторские права должны быть отражены в списке;

- 5.8. Выполнить развертку, сборку и дистрибуцию при необходимости.

1.6 Аналоги

Разрабатываемая система имеет некоторое количество аналогов, но ни один из таковых не обладает свойствами и функционалом, присущими проектируемой и разрабатываемой системе. Она абсолютно точно будет иметь

свою собственную аудиторию и будет иметь достаточное пространство на рынке для улучшения опыта пользователя, увеличения контента и захвата новой аудитории.

В настоящих условиях на рынке проект будет иметь свою нишу и, при условии достойной реализации, так или иначе будет прибыльным и востребованным.

Большая часть аналогов являются продуктом, созданным для менеджмента социальных сетей со стороны SMM и подобной работы в сфере PR и маркетинга. Иначе говоря, такие менеджеры созданы не для обычного человека, а для человека, который использует соцсети для работы, используя платформу как метод распространения и сбора информации, а не для повседневного общения. То есть, грубо говоря, злоупотребляет технологией для выгоды своей компании или информирования аудитории, так как первоначальная задумка соцсетей — общение. Нарушение первоначальной задумки — следствие динамики развития технологий и человечества. Сегодня эта изначальная задумка отошла на второй план.

Системы, упомянутые далее, не являются плохими по качеству или неправильными по концепту, как могло бы показаться. Они выполняют свою цель, ради которой разработаны. Такие системы выполняют поставленные цели с разной степенью успешности. Имеется в виду, что функционал схож, цель подобна, и, в конечном итоге, разница заключается в клиенте и в его требованиях, которая в финальную очередь сводится к удобству конкретного клиента. Система «для всех» не будет работать ни при каких условиях, так как сегодня высоко ценятся производительность, минимализм, простота использования, а также цена и надежность. Достижение всего из указанного нерелевантно.

Аналоги для SMM, PR, FAQ и поддержки:

1. Hootsuite (<https://www.hootsuite.com/>);
2. Sprout Social (<https://sproutsocial.com/>);
3. Emplifi (https://emplifi.io/?utm_source=socialbakers.com);

Хотя подобные системы и располагают функционалом для достижения практических целей разрабатываемого проекта, такие системы не нацелены на обычных людей, тем самым и приоритет реализованных функций не совпадает с желаниями и требованиями клиентской базы, то есть с необходимостями повседневных пользователей. Таким образом, не достигаются социальные, глобальные, частные цели. Также следует учесть, что такие системы усложнены и предоставляют излишние функции, заполняя интерфейс и усложняя обращение. Ключевыми аспектами таких систем является отслеживание значений, статистики для автоматизации ответов и настройки внешнего единого корпоративного вида страниц.

Обобщая всё вышесказанное, данные аналоги нацелены на управление страницами компаний, например «ВК», мероприятий, например концертов, продуктов, например велосипедов. Содержание таких систем предоставляет скорее управленческие, SMM, data science, big data, PR функции, а не коммуникативные и социальные. Вряд ли хотя бы один из аналогов заинтересует и устроит обычного пользователя соцсети.

Аналоги для блогеров, медиа личностей, известностей:

1. Buffer (<https://buffer.com/>);
2. Falcon.io (Brandwatch) (<https://www.brandwatch.com/>);
3. Agorapulse (<https://www.agorapulse.com/>);

Вторая часть аналогов направлена на блогеров и прочих медиаличностей, использующих такие сети в качестве методов заработка и/или распространения собственного контента. Такие системы могут быть использованы обычными пользователями, но они снова не располагают необходимыми качествами. Функции таких разработок заключены в предотвращении спама, в автоматизации рассылок, в упрощении коммуникации с большим количеством собеседников, в первую очередь. Подписчики и собеседники крупных медиаличностей исчисляются тысячами, а иногда миллионами. В этом и цель таких систем — упрощения общения знаменитости с пользователями и сокращения влияния негативного эффекта

пользователей. Системы предоставляют дополнительный функционал в виде доступа из разных мест, интегрированной или сторонней модерации, возможности заходить нескольким лицам на один аккаунт. По этим причинам разрабатываемое ПО имеет гораздо бóльшую релевантность в своей нише.

В общем и целом, системы управления соцсетями для знаменитостей и для продуктов схожи, но зачастую слишком фундаментальны и усложнены. Иногда требуется дополнительное обучение для их использования. Они предоставляют преимущественно полезный функционал с преимущественно хорошей реализацией, но сложно и много не всегда хорошо. Разрабатываемая система проста, элегантна и идеально подходит для обычного пользователя тем, что она предоставляет именно тот функционал, что ему требуется, то есть функционал агрегирования данных и приведения социальных сетей к одному узлу, одному приложению, действуя по правилу «всё в одном».

Стоит также отметить, что в свете культуры отмены со стороны западных государств, ответных мер РФ и проектов защиты персональных данных количество подобных систем, доступных пользователю из России, сократилось в разы. Недостаток предложения на рынке провоцирует сильный рост спроса. В таких условиях создание новой системы с данным функционалом востребовано, и в конечном итоге это означает, что такая система не только быстро и уверенно найдет свою аудиторию, но и будет интересна инвесторам, государству и в целом пойдет на пользу населению.

Подводя итог, можно смело заявить, что подобная система является актуальной, релевантной, востребованной, нишевой, актуальной и необходимой. Она найдет свою аудиторию, так как имеет большой спрос. У нее нет прямых конкурентов, что поможет без больших затрат и с малым темпом развивать функционал. Такая система будет интересна народу, государству, инвесторам. Система будет стремиться решить две глобальных проблемы человечества: люди будут проводить меньше времени в социальных сетях, но будут извлекать гораздо больше из этого пользы. В целом, можно заявить, что подобный концепт стоит проработки, разработки, монетизации и

дальнейшего развития, так как имеет не лимитированный спрос. Пока человек общается, используя соцсети, данная система будет стоить своей разработки и поддержки.

1.7 Анализ области и объекта применения

Общество, социум (от лат. *societas* — общность, союз, от лат. *sociāre* — объединять) — это совокупность людей, объединенных определенными общественными взаимосвязями, институтами, ценностями, обычаями и правилами, которые регулируют их взаимодействие и поведение в рамках определенного контекста[32]. Это сфера, где люди живут, работают, общаются и сотрудничают друг с другом. В обществе существуют различные социальные группы, институты, нормы и ценности, которые формируют общественную жизнь.

Человек в любом его контексте и в любом контексте общества является его неотъемлемой частью. В любом случае он участвует в его процессах и является его неотделимым элементом. В рамках социума действуют сотни тысяч правил и закономерностей, феноменов исследованных и нет, также в нем участвует все население мира. Разработка в данной сфере сопряжена с рисками, сложностями и неоднозначностями.

Для дальнейшего анализа и обоснования определим элементы взаимодействия и создадим список используемых терминов:

1. Объектом анализа и исследуемой структурой будет размытое понятие общества. Под ним будем в первую очередь подразумевать группу людей, взаимодействующих друг с другом, охватывающую все население Земли;
2. Субъектом анализа и исследуемой структуры будет каждый элемент, который входит в общество. В первую очередь, это личности, люди.

Теперь, когда мы имеем четкий набор используемых определений и базовое определение общества, мы можем выдвинуть предположение об

особенностях. Социум имеет следующий набор главных особенностей, частично интерпретация [\[16\]](#), частично собственный анализ с чистого листа:

- Стохастичность — из-за огромного количества параметров и уникальных субъектов все процессы в обществе имеют стохастический характер. В этом контексте стохастичность определяется присутствием случайности в процессах из-за невозможности учета всех параметров и субъектов;
- Динамизм — процессы в обществе не регулируются, а являются по своей природе итогом социального, технологического прогресса. Общество в этом плане подобно эволюции, оно зависит не от действий какого-либо человека, а от закономерностей, поступков и решений человечества. Высокий динамизм и неконтролируемость также связаны с трансцендентностью самого понятия и внутренних процессов;
- Многогранность — каждый субъект общества, человек, имеет свой уникальный набор характеристик, повторений нет, что делает общество многогранным;
- Глобальность — общество само по себе, определение всеохватывающее, что значит, что любой человек в мире является его субъектом. Значит, что общество объединяет всех людей на Земле;
- Неизмеримость — не существует точного метода измерения процессов и субъектов общества. Следствие из глобальности и стохастичности;
- Неоднозначность — не существует четкого определения для любого из элементов общества. Следствие из неизмеримости;
- Структурная инерция — единственный способ изменить общество, это менять его целым обществом. Иначе говоря, один человек не сможет ничего изменить, необходим каждый субъект для подобного изменения. Следствие из глобальности и динамизма;
- Неповторимость, неуправляемость, трансцендентность и независимость. По крайней мере, до определенного этапа развития человечества, или навсегда. Нет способа повторить, имитировать в определенной степени

общество, понять и обосновать его процессы однозначно, управлять однозначно. Следствие из неизмеримости, неоднозначности, стохастичности. В отличие от технических процессов, где Точка Омега еще не достигнута, в социуме по своей специфике она преодолена в моменте появления общества.

Вследствие этих факторов контролируемая работа с обществом и его процессами затруднена и не поддается точной оценке. В таких условиях любая технология и методология, задействованная в сфере, будет иметь Эффект Бабочки, ее реализация будет в определенной мере случайна. Изменение любого масштаба может привести или не привести к серьезнейшим изменениям в целом. Поэтому разработка в данной области имеет несколько неопределенный характер, а также может косвенно повлечь ряд других неожиданных изменений.

Воробьева И.А. и Костерев Р.А.: «Однако перемены, вызванные научно-техническим прогрессом, достаточно сложно прогнозируемы и требуют значительных усилий для анализа и объяснения. «Традиционная теория» не всегда способна объяснять и давать четкие ответы на надвигающиеся вызовы, тем самым поднимая проблему актуальности современной социологической теории в понимании цифровой структуры общества»[\[6\]](#)

Социальные сети и цифровые коммуникации являются как раз теми разработками, которые в свое время имели данное свойство и коренным образом изменили то, как сегодня происходит общение. В данных конкретных условиях сегодня сложно представить общение без социальных сетей, в то же время жить без цифровых коммуникаций сегодня тяжело и не принято.

В контексте разработки приходится работать с указанными особенностями, учитывать необходимо каждую из них. Из-за этого разработка в социальной предметной сфере сопряжена с определенными сложностями.

Разберем каждую из особенностей общества при условии проектирования ПО в упомянутой сфере.

Стохастичность, очевидно, добавляет случайность в разработку, необходимо иметь в виду, что социум имеет данное свойство и подстраивается под условия. Например, можно разрабатывать и далее улучшать малые подсистемы и смотреть, как в целом ведет себя главная система, тем самым нивелируя некоторую долю рисков. Иначе можно целенаправленно замедлять разработку для своевременного анализа результата. Однако в любом случае предугадать всё и убрать все риски из уравнения проектирования невозможно, и это надо учитывать. Стохастичность социума имеет замедленный ударный характер, то есть зарождение, появление и достижение апогея стохастического события происходит в начале медленно, а затем убыстряется, достигая неконтролируемости к своему пику. В таких условиях всё, что остается, — это предупреждение и заведомый анализ причин стохастического события и попытка исправить событие до его завершения. Что на самом деле является квазивозможным.

Динамизм социума подразумевает слабо отслеживаемое, импульсивное, по сути, неконтролируемое развитие и изменение структуры и содержания. Опять же, в ходе разработки ПО данной сферы главное — вовремя подстраиваться под новую конкретику. Динамизм определяет социум как объект, движимый действиями субъектов. Иначе говоря, система общества всегда в движении, прогрессе, регрессе, в целом в изменении, так как каждый ее субъект ежесекундно и постоянно вносит свои малозначимые для себя, но в массе критически значимые изменения для системы. Динамизм социум имеет низкую скорость, по этой причине при разработке ПО никаких дополнительных изменений вносить не имеет смысла.

Многогранность идет вместе с динамизмом. Без многогранности динамизм такой неточной, трансцендентной системы, как общество, невозможен, ведь уникальность каждого субъекта определяет уникальность их взаимосвязей и взаимоотношений. В таком контексте эта уникальность определяет общий вектор развития объекта. То есть уникальность субъектов порождает уникальность объекта. Многогранность усложняет нацеленную

разработку ПО, в таких условиях сложно разрабатывать систему под кого-то, легче разрабатывать под всех, принимая во внимание некий средний случай.

Глобальность, как и многогранность, определяет всеобъемлемость социума. Наряду с другими аспектами эта характеристика создает полный и всеохватывающий динамизм. Глобальность — априори особенность социума, так как не существует субъекта вне грани объекта. В конечном итоге это значит, что каждая личность так или иначе воздействует на объект. Единственный контекст, исключаящий личность из общества, — тотальная изолированность человека от социума, что со своей стороны ставит вопросы об осознании и идентифицируемости конкретного человека как личности. Глобальность мало влияет на проектирование систем.

Неизмеримость и неоднозначность, сопутствующие глобальности особенности, неизмеримость и неоднозначность общества — следствие уникальности каждого из его субъектов. При таких условиях нормализация и формализация объекта невозможна, так как каждый субъект неповторим во всех смыслах, а формализация и объективизация всего субъективизма субъектов не представляется возможным. Особенность накладывает определенные сложности с проектированием систем. Отсутствие возможности анализа и четкой оценки, категоризации и структуризации субъектов в системе — клиентов — делает систему несколько полагающейся на удачу. Доля этой удачи будет мала, так как в целом средний случай в обществе относительно точно отражает достаточное количество параметров для работы.

Структурная инерция общества наряду с динамизмом добавляет Эффект Бабочки любому проектируемому в сфере ПО. Любая система, сделанная для работы с обществом, может иметь критические последствия для всего объекта или не иметь эффекта вообще. При малейшем изменении в процессах между субъектами, при условии большого охвата, то есть большого количества субъектов, можно наблюдать каскадный (лавинный) эффект. Лавинный эффект подразумевает малейшее изменение, которое влечет за собой все более крупные изменения при дальнейшей работе системы.

Неповторимость, неуправляемость, трансцендентность, независимость. Главные сложности разработки в любых временных рамках. Эти особенности вместе взятые могут за одну ночь определить вашу систему как безыдейную, бессмысленную, нерелевантную и так далее. Также благодаря этим факторам любая система, спроектированная для общества, рано или поздно вымирает. Имеется в виду не конкретная разработка, а сама идея разработки. Так, например, сегодня можно наблюдать вымирание общения с помощью почты, а также вспомогательной системы — почтового ящика. В таких условиях необходимо понимать, что это естественный процесс всего в мире. И время — единственное, что важно. Однако в контексте коммуникаций и бизнеса это имеет более критическую и менее философскую роль, ведь в системах цифровых коммуникаций в первую очередь важны актуальность. Системы не имеют смысла, если хотя бы группа людей их не использует. В отличие от систем в экономике, экологии, инженерии, менеджменте, где технологии могут применяться даже без ведома людей, например система бронирования, которая находится в использовании несколько десятилетий, потеряв всякий элемент эффективности и корректности.

Заключая всё вышесказанное, можно сказать, что при проектировании и разработке ПО в социальной сфере нужно следовать нескольким методическим рекомендациям. Следует учитывать все особенности социума.

Модульность системы важна, необходимо делить систему на малые подсистемы, далее тестировать эффективность работы и эффект на социум и объединять в более крупные структуры. Имеет смысл целенаправленно замедлять разработку при условиях введения новых вариантов использования. Необходимо анализировать эффективность работы постоянно и принимать решения по предупреждению малейших отклонений. У общества низкая скорость динамизма, придется соперничать в скорости только с конкурентом. Система ни в коем образе не должна фокусироваться на исполнении желания конкретного человека, она должна следовать общим тенденциям общества. Это идет вразрез с целью разрабатываемой системы и заявленной

актуальностью, что еще сильнее усложняет разработку. Глобальность добавляет системе безграничный потенциальный спрос, пока все люди не будут ей пользоваться, будет возможность найти новую аудиторию. Четко выверенный средний случай повысит эффективность системы в свете невозможности сравнивать и структурировать субъекты. Полноценная формализация субъекта невозможна, в любом случае придется работать с рисками, связанными со случайностью и неопределенностью. Следует тщательно выбирать, планировать и реализовывать новый функционал, общество обладает структурной инертностью. При условии устаревания идеи коммуникации и/или взаимодействия систему можно свертывать и архивировать. Так или иначе, рано или поздно это произойдет, надо быть готовым к тому, что в любой день в свете социальных изменений система придет в полную негодность. Выборка и частичная интерпретация [\[29\]](#).

Цифровые коммуникации — это процессы обмена информацией, осуществляемые с помощью цифровых технологий, таких как интернет, мобильные устройства, социальные сети и другие электронные средства связи. С точки зрения социологии, цифровые коммуникации рассматриваются как важный аспект современных социальных взаимодействий и структур.

Цифровые коммуникации стали возможны благодаря научно-техническому прогрессу человечества в течение информационного и атомного века развития. Разработки в сфере телекоммуникаций, телефонии, интернета и мобильной связи позволили изменить строй общества и принципы взаимодействия в нем. Главным образом, изменилась скорость общения, эмоциональная привязанность каждого из участников, доступность и свобода [\[19\]](#).

Цифровые коммуникации не являются полностью автономным и независимым видом, наоборот, термин «цифровые» в данном случае отражает среду взаимодействия, а не конечный принцип. Однако сегодня, с развитием технологий и, в частности, цифровых коммуникаций, они уже частично приобрели необходимый уровень развития и комплексности, когда о них

можно судить как о наследнике, о кумулятивном продолжении развития коммуникаций. Что в конечном итоге значит, что, когда мы говорим о цифровых коммуникациях, мы говорим не только лишь о среде, но и о сопутствующих особенностях, привнесенных новой средой в традиционные коммуникации.

Для подобного анализа приведем набор терминов для лучшей навигации в объекте исследования:

1. Объект анализа в данном случае — коммуникация, процесс обмена информацией между личностями в контексте общества. В данном случае — цифровая коммуникация;
2. Агент анализа — стороны коммуникации, люди, которые осуществляют обмен данными. Каждый агент цифровой коммуникации либо принимает входящую информацию, либо ее отсылает.

Разберем цифровые коммуникации с точки зрения их особенностей, преимуществ и недостатков относительно традиционного личного общения. Будем рассматривать как можно более общий случай цифровых коммуникаций и постараемся не уходить в крайности. Цифровые коммуникации имеют следующий набор особенностей, частичный анализ [\[16\]](#):

- Виртуализация. Изменение специфики общения в сторону цифровой. Нет необходимости лично взаимодействовать друг с другом. Всё общение может выполняться дистанционно внутри информационных структур;
- Децентрализация. Цифровые коммуникации — итог научно-технического прогресса и реалий информатики. При таких условиях, благодаря множеству факторов и частичному совпадению развития, пошло в сторону децентрализации и распределенности, а не в сторону централизации, это относится именно к интернету. Благодаря этому в цифровых коммуникациях объект находится конкретно между агентами без участия некоторого координационного, операционного центра. Этот аспект оспариваем, но в целом описывает направление и

специфику развития. Следствие виртуализации. Следует отметить, что в соцсетях картина обратная[9];

- Гибкость. Цифровая коммуникация предлагает широкий спектр видов разных видов общения. Тем самым предлагая новые горизонты для коммуникации между людьми. Многообразие видов улучшает гибкость. Следствие из виртуализации, децентрализации;
- Конструируемость, социальная абстрактность. В отличие от традиционного общения, цифровое располагает набором методов и функций для контролируемой структуризации общества. Благодаря этому появляется возможность создания виртуальных групп и организации социума по интересам. Социальная абстрактность же улучшает обособленность общества, создавая еще более мнимый конструкт для общения. Такой конструкт будет иметь гораздо более точную и формализованную структуру, нежели конструкты традиционных коммуникаций;
- Неоднозначная, нефиксированная идентифицируемость. Человек, участвующий в цифровых коммуникациях, имеет возможность видоизменять свой социально идентифицируемый образ. Врать, изменять, адаптировать свои черты ради какой-либо цели. Следствие из конструируемости, социальной абстрактности, виртуализации;
- Доступность и свобода. Цифровые коммуникации доступны любому человеку, имеющему доступ к интернету или другим технологиям цифровизированной коммуникации. Этот пункт можно оспорить в свете регулировок цифровых коммуникаций ведомствами разных государств, но в целом картина определяет их как свободные и доступные. У интернета по большей части нет национальности и политики, по этой причине любой человек с земного шара может связаться с другим человеком. Следствие неоднозначной идентифицируемости, децентрализации, гибкости;

- Плюрализм мнений. Наличие возможности изменять идентифицируемость и адаптировать свою персоналию по желанию, а также благодаря децентрализации и доступности, но в условиях глобализации порождают четкую систему отображения плюрализма мнения. Такая система будет в максимальной степени отражать множественность и свободу мнений. Следствие доступности, нефиксированной идентифицируемости, децентрализации;
- Смещение границ частного и публичного. Исчерпывающая особенность, следствие из нефиксированной идентифицируемости, свободы, социальной абстракции. Из-за этих особенностей количество предлагаемой информации о личности имеет большую амплитуду, чем в традиционных коммуникациях;
- Трансформации временных и пространственных барьеров. Из-за изменения принципов общения временные и пространственные барьеры терпят ощутимые изменения. Прямое следствие — из гибкости и виртуализации. Благодаря новым методам общения теряется и трансформируется скорость и место общения[\[23\]](#);
- Эволюция социокультурных норм. Очень неоднозначная и самая сложно оцениваемая особенность цифровых коммуникаций. Суждения про данную характеристику крайне субъективны и относятся к социологическим исследованиям высшего порядка, имеющим свой семантический вес только в условиях исключительного профессионализма в сфере. В общем же плане подразумевает изменения социальных принципов, институтов, процессов под сильным давлением и влиянием новых видов коммуникаций. Эволюция социологической и культурных сфер в заведомо неоднозначном векторе. Следствие из всех особенностей;
- Нравственные, структурные, фундаментальные изменения социума. Имеет такую же специфику, как и предыдущая особенность;

Добринская Е.Д.: «Таким образом, функционирование коммуникационных сетей создает сетевое измерение цифрового общества. Это сетевое измерение отражает его супер связанный характер, который приводит к таким социальным последствиям, как усиливающийся контроль, нарушение привычных границ приватного и публичного, рост социального неравенства и многое другое.»[10]

Воробьева И.А. и Костерев Р.А.: «Таким образом, качественно новыми характеристиками современного общества, свидетельствующими о наступлении эпохи цифрового общества, являются мобильность, процессы цифровизации и сетевизации, усложнение социальной системы. Цифровое общество представляет собой инфраструктуру, функционирующую посредством цифровых информационно-коммуникационных технологий, которая основывается на сетевом взаимодействии между пользователями системы.»[6]

Заключая всё вышесказанное, эти особенности усложняют или упрощают разработку в сфере цифровых коммуникаций, при этом изменяя принципы и процессы на различных этапах разработки и проектирования.

1.8 Анализ разрабатываемой системы

При разработке ПО в рамках цифровых коммуникаций и соцсетей следует учитывать связность сети, количество соединений между разными пользователями. Количество таких соединений диктует актуальность и эффективность той или иной социологической системы. Следует принимать в расчет, что разные связи определяют разную результативность и узла, и в целом соцсети. Слабые связи, редко используемые, гораздо более важны, чем сильные связи[21]. Это надо учитывать при разработке ПО в данной сфере. Данная система разрабатывается как раз ради усиления этих слабых связей, что в конечном итоге усиливает соцсети в их собственной задумке.

В целях достижения большой эффективности системы следует понимать, как работают социальные сети, не столько даже в техническом аспекте, хотя это тоже важно, сколько в концептуальном плане.

На примере VKontakte API[\[40\]](#) можно разобрать, по какому принципу работают социальные сети с технической и идейной точки зрения[\[5\]](#). Можно отобрать и разобрать методы, реализуемые для получения доступа к данным и их обработки. При небольшом углублении в тематику становится очевидно о природной, неконтролируемой структуре взаимосвязей в социальной сети. В то же время, можно провести корреляцию этой структуры, например, с исконно природной иерархией муравьев, пчел и других роевых существ с четкой структурой общества.

Простота работы с API социальных сетей иногда доходит до абсурда, методы напрямую описаны и напрямую возвращают требуемые данные без лишних условий. С другой стороны, разработка систем не так проста в контексте создания комплексных структур с масштабируемостью, множественным доступом и работой с аутентификацией. Также не следует выпускать из виду юридическую сторону вопроса.

Параллельно с анализом постепенно приходит понимание, что на бумаге в идеальном мире концепт разрабатываемой системы идеален, так как закрывает слабые места социальных сетей, и параллельно не лишая пользователя их слабых сторон. Немного философская статья [\[11\]](#) как раз описывает высшую цель разрабатываемой системы. Описывая, немного максималистично, идеологически, то, к чему должно стремиться общение в обществе. К сожалению, с момента написания (2011 г.) человечество не только не приблизилось к новой идее цифровых коммуникаций, а, наоборот, отделилось еще дальше.

Вернемся к социальным сетям. Серия статей [\[28\]\[15\]\[21\]\[24\]](#), подробно описывает принцип их работы и точки соприкосновения, те места, которые можно использовать для взаимодействия и улучшения доступных на данный момент элементов. Сообщества социальной сети — главные элементы

соцсетей, они образуют группы разных людей, объединенных каким-либо фокусом (интересом). Любая соцсеть строится на слабых связках между многочисленными сообществами, которые держатся на сильных связках.

Слабая связка — редко используемая связка между двумя агентами социальных сетей. Они формируют минимальное остовное дерево любой социальной сети и поэтому критичны для её работоспособности. Их мало, но они очень длинные, объединяют очень разных, далеких друг от друга людей.

Сильная связка — часто используемая связка между двумя агентами социальных сетей. Они формируют групповые связи внутри сообщества, группы. Сильные связки множественны, но они короткие, то есть очень тесные. По причине чрезмерной неустойчивости слабых связок, сегодняшняя ситуация на рынке и в целом ситуация с утилизацией соцсетей неидеальная.

В разрабатываемой системе главная идея заключается как раз в усилении слабых связок, тем самым предоставляя усиление минимального остовного древа соцсети. Заклучая всё сказанное, система в разработке будет эффективна не столько в общении между близкими людьми, сколько в напоминании о возможности общения с разными знакомыми.

Процесс образования связок — кропотливый алгоритм, требующий наличия и инициации диалога. Сначала создается слабая связка, потом при достаточной релевантности контакта для каждого из агентов коммуникации она становится сильнее. Таким образом, в разрабатываемая система может участвовать в создании новых сильных и слабых связок. Участники сильных связок рано или поздно будут переходить к нативной соцсети в браузере, но в то же время они будут создавать слабые связки с новыми контактами в приложении.

Утилизация нескольких социальных сетей — очень хорошая черта, позволяя увеличивать число связок экспоненциально. Данная система со стороны анализа сильно увеличивает количество слабых связок, при этом не отменяя возможность продолжать поддерживать сильные между пользователями.

2 ПРОЕКТИРОВАНИЕ

2.1 Проектирование решения

В целях достижения итоговой задачи проекта необходимо спроектировать и далее разработать технологию прикладного характера, которая решает поставленные проблемы в той или иной степени. Разработка без проектирования возможна, но будет крайне неэффективной и приобретать недостатки на всех этапах процесса. Эти недостатки, в начале имеющие низкий приоритет и слабое влияние на систему в общем, в конце будут критическими и поставят под угрозу всю разработку программного обеспечения. Параллельно подобные недостатки будут увеличивать затраты различных ресурсов на разработку, включая в первую очередь время и финансы. Чтобы избежать этого, необходимо перед и затем в ходе разработки проектировать систему, чтобы добиться лучшей эффективности производственного процесса. Параллельно проектированию данной системы будем ссылаться и рассматривать [\[20\]](#), частично как референс, частично для понимания принципа работы соцсетей и их API — Application Programming Interface.

Исходя из цели разработки, начнем проектировать решение, то есть формализовать, структурировать разрабатываемую систему, придавая ей вид теоретически возможной.

Согласно правилам и общим принципам проектирования ПО, начнем с того, что конкретно нам надо реализовать. В этих условиях будем отталкиваться от цели и задач разработки. Перечислим их еще раз, преобразуем к более прикладному характеру.

- Цель работы — разработка программного обеспечения, нацеленного на агрегацию содержания социальных сетей пользователя;

Цель работы выражена через главные функции разрабатываемой системы. То есть через следующие упомянутые главные функции.

1. Пользователи могут получать и просматривать сообщения из разных источников (соцсетей) от разных пользователей (клиентов);
 - а. Пользователи могут просматривать приложения к сообщениям;
2. Пользователи могут составлять и посылать сообщения с одной платформы любому пользователю через любую доступную (отправителю и получателю) социальную сеть;
 - а. Пользователи могут добавлять приложения к сообщениям;
3. Пользователи могут подключать (активировать агрегацию) разные доступные ему соцсети;
4. Пользователю доступен набор настроек для персонализации, безопасности, доступности, сетевого взаимодействия и прочих общих аспектов программы;

Функции под пунктами 1, 2, 3 реализуются с условием взаимодействия с API, ПО третьих лиц. Иначе говоря, используют другую систему как черный ящик для получения или отправки данных.

Приведем эти функции к формату вариантов использования для уточнения и формализации разработки с точки зрения требуемого функционала. Объединяя все функции с целями и потенциальным вариантом реализации, мы можем составить UML — Unified Model Language — use case диаграмму, которая будет отражать варианты использования.

На диаграмме [\[Приложение А\]](#) изображена UML вариантов использования, которая включает в себе требуемый функционал. На данной диаграмме присутствуют три условности для более качественного отражения концепта системы. В первую очередь, четкое разграничение между разрабатываемой системой и агрегируемой системой. Это сделано для четкого обозначения того, что отчасти варианты использования исполнены не в рамках системы, а в рамках использованного ПО третьей стороны. Во вторую очередь, в проекте присутствуют четыре варианта использования, описанные через включение, которые как раз исходят из рамок одной системы и входят в рамки другой, на их месте подразумевается работа через API (в плане системы) или

может быть исполнена лично (в плане человека, использующего соцсети), опять же, это сделано для наглядности принципа работы. В третью очередь, это отношение зависимости агента — пользователя приложения от агента — пользователя социальных сетей, для того чтобы показать, что действия агентов по сути идентичны, но выполняются с разных систем. Главное достоинство системы — как раз тот факт, что пользователь может пользоваться только одной системой в обмен на использование нескольких социальных сетей.

Эта диаграмма и в общем описанный функционал покрывает первичную версию продукта, то есть не определяет всё, на что способна система.

Исходя из листинга вариантов использования, переходим к следующей стадии. Создание перечня подсистем. На данном этапе мы можем определить набор подсистем для более точной и эффективной реализации функций. Переведем перечень подсистем, исходя из вариантов использования.

- **Подсистема интегрирования, подключения соцсетей.** Будет ответственна за добавление новых и удаление соцсетей. Ключевой элемент, который позволяет собирать данные и с разных источников;
- **Подсистема агрегации контактов.** Будет ответственна за обновление списка контактов пользователя системой. Список контактов должен составляться с использованием всех интегрированных соцсетей и, потенциально, должен сливать контакты из разных соцсетей одного и того же человека в один;
- **Подсистема агрегации сообщений.** Ответственна за обновления списка сообщений определенного контакта пользователя системой. Для каждого контакта в системе агрегация сообщений позволяет получить список и состав всех сообщений этого контакта;
- **Подсистема составления и отправки сообщений.** Главной задачей данной подсистемы будет являться составление сообщения с использованием интерфейса системы и отправка сообщения с использованием API соцсети;

- **Подсистема конфигурации приложения.** Разнообразные настройки для персонализации, фильтрации, сети.

Другие подсистемы не обозначены по причинам некритичности, слишком маленького масштаба для описания, неописываемости характеристик и общей надобности, также они могут иметь вспомогательную или дополнительную специфику. В конечном итоге описанных подсистем уже достаточно для подробного описания, проектирования системы.

Переходим к следующему этапу проектирования, теперь, когда у нас есть набор вариантов использования и список подсистем, мы можем переходить к проектированию классовых отношений и модулей. Для этого обратимся к UML class diagram для отображения взаимоотношений классов.

На диаграмме [\[Приложение Б\]](#) изображена диаграмма классов, которая отображает взаимосвязи между классами. С первого взгляда перегруженная диаграмма покрывает около 70–80% процентом проектируемой системы. Рассмотрим ее более подробно. Ключевыми, центровыми классами остаются классы сообщения (Message), контактов (Contact), приложения (Application). Стоит отметить, что данная схема имеет ряд условностей для лучшего отображения ее элементов, в то же время необходимо заметить, что конечная реализация в частностях может отличаться от итоговой и этапов проектирования. В первую очередь это касается полей интерфейсов с разными модификаторами доступа, они отображают модификаторы доступа для имплементаций. Так как сами имплементации на схеме не описаны. Вся система строится на взаимодействии и работе каждого из ключевых классов.

- **Интерфейс сообщений.** Заключает в себе всю необходимую информацию о каждом конкретном сообщении. Описывает абстрактную структуру сущности сообщения. Имплементируется для каждой социальной сети отдельно;
 - *Данные*, желательно в бинарном коде, — само сообщение;
 - *Флаг* — отображение входящего или исходящего сообщения, а также для отметок приложений и прочей информации;

- *Время* — композиция к классу временных отметок;
 - *Приложения* — агрегация к классу приложений;
 - Набор функций для взаимодействия с полями: *Getter* для данных; *Getter* для времени; *Getter* для флага; *Getter* для приложений; *Добавить приложение*; *Перевести все сообщения в бинарный вид*, включая приложения, время, флаг;
 - **Класс приложений.** Заключает в себе информацию о каждом конкретном приложении для каждого/любого из сообщений;
 - *Ссылка на сообщение*, к которому прикрепляется это приложение;
 - *Флаг типа приложения*;
 - *Размер приложения*;
 - *Название*;
 - Набор функций для взаимодействия с полями: *Getter* для флага; *Getter* для размера; *Getter* для ссылки на сообщение; *Getter* для сообщения; *Getter* для хранящихся данных; *Getter* для спецификации приложения;
 - **Класс приложений перегружается** в зависимости от типа приложения, в таком случае класс может приобретать следующие поля;
 - *Размер приложения (для всех)*;
 - *Длина приложения (для аудио, видео)*;
 - *Формат приложения (для всех)*;
 - *Кодек приложения (для видео, изображения)*;
 - *Кодировка приложения (для документа)*;
 - *Перегруженный Getter для данных (для всех)*;
- Это поверхностное проектирование приложений, и данная структура является лишь примером, шаблоном для будущего расширения.
- **Класс временных отметок** (необязательный) может быть заменен библиотеками и внутренним функционалом фреймворка. Включает в

себя поля: *секунда, минута, час, день, месяц, год* и *функцию, возвращающую контейнерную дату*;

- **Интерфейс контактов.** Один из ключевых классов, который хранит в себе информацию обо всех доступных клиенту контактах. Является важнейшим классом системы наряду с классом сообщения, так как определяет работу системы в целом. Имплементируется для каждой социальной сети отдельно;

- *Имя контакта*;
- *Социальная сеть* — ссылается на объект класса социальных сетей;
- *Словарь сообщений*, со строкой состояния, флагом состояния, временной отметкой, сообщением;
- *Временная отметка*, последний раз в сети;
- *Временная отметка*, последнего сообщения;
- Набор функций для взаимодействия с другими классами: *Getter* всех сообщений; *Getter* для последнего сообщения; *Getter* для сообщения по индексу; *Setter* для установки имени контакта; *Getter* для получения объекта социальной сети; *Setter* для установки текста и приложений сообщения; *Creator* для создания нового сообщения внутри контакта; *Sender* для отправки сообщения;

- **Интерфейс социальных сетей**, отображающий доступную социальную сеть, является отправной точкой агрегирования, критический класс для работы агрегирования. Имплементируется для каждой социальной сети отдельно;

- *Название*;
- *Список контактов*, агрегация контактов, в этом отношении данное поле является главным с точки зрения концепта разрабатываемой системы;
- *Метод для получения списка контактов*;

- **Класс приложения** (управляющий, главный) управляет работой приложения, его визуальной составляющей, соединяет воедино все подсистемы. Критический для работы системы класс;
 - *Лист контактов*, все контакты всех социальных сетей, использует метод внутри класса социальных сетей;
 - *Список социальных сетей*, агрегация к классу социальных сетей;
 - *Список всех активных сообщений* заполняется только сообщениями выбранного контакта, использует метод внутри класса контактов, туннелированное использование, класс социальных сетей не задействован;
 - *Локальные настройки* — агрегация в сторону класса настроек.
 - Набор методов: *Getter* для всех социальных сетей, *Getter* для всех контактов, *Getter* для всех сообщений одного контакта, *Creator* для создания сообщения создает объекта класса сообщение, туннелированное использование класса контактов, *Sender* для отправки созданного сообщения, использует *Creator*, *Updater* для обновления внешнего вида окна, *Fetcher* для получения новых сообщений, туннелированное использование объекта контакта, *Fetcher* для получения данных из базы данных и файлов конфигурации, использует менеджер хранящихся данных;
- **Класс настроек**, включает в себе словарь отношений ключ, значения и методы для его манипулирования, использует API для доступа к БД — базам данных;
- **Класс менеджера хранящихся данных** для хранения кэша, контактов, сообщений и т. д. для более быстрой загрузки. Используется только для данных социальных сетей. Данные приложения используют API к БД конфигурации напрямую;
 - Тип хранимых данных;
 - Хранимые данные, агрегация к классу хранимых данных;
 - *Getter*, *Setter* для хранимых данных;

- **Интерфейс хранимых данных** для контейнирования бинарных данных и учитывания их типа;
- **DBM** — Database Manager — на схеме это встроенный в СУБД — систему управления базами данных — и/или ЯП — язык программирования — инструментарий для управления БД;
- **Any*** — на схеме подразумевает замещение в любом количестве, любой социальной сетью;

Теперь, когда у нас есть UML-диаграмма классов, мы можем идти дальше. На основании данной диаграммы можно уже приступить к разработке, но доведем проектирование до конца и покроем наше решение целиком, чтобы в ходе разработки не возникали вопросы и не приходилось возвращаться к проектированию. В ходе проектирования классов были обозначены используемые БД и СУБД для более точной трактовки данных элементов системы.

Переходим к следующим этапам проектирования. На данный момент мы имеем список вариантов использования, диаграмму вариантов использования и диаграмму классов с расшифровками ее элементов. Рассмотрим два главных варианта использования для более точного исполнения в ходе разработки.

Первый из ключевых является критический вариант использования «просмотреть беседы», вариант использования 1.1 в [\[Приложении В\]](#). Этот вариант подразумевает возможность использования приложения, при котором пользователь может просматривать список сообщений из разных источников. Всего рассмотрено три процесса пользователя: получить все сообщения от всех контактов (стандартное поведение); получить все сообщения от контакта X; получить сообщения Y от контакта X (специальное поведение). Все три способа описаны на уровне взаимодействия человека — разрабатываемой системы, разрабатываемой системы — агрегируемой системы, агрегируемой системы — внутреннего потенциального контакта.

Второй из ключевых является критический вариант использования «написать сообщения», вариант использования 2.1 в [\[Приложении Г\]](#). Этот

вариант подразумевает возможность использования приложения, при котором пользователь может составлять и отправлять сообщения в разные соцсети. Всего рассмотрен один процесс пользователя, который описан на тех же уровнях, что и в варианте использования 1.1. Этот процесс более комплексный, так как, во-первых, использует часть из варианта использования 1.1, в какой-то степени можно заявить, что 1.1 является частью 2.1, во-вторых, описывает больше ситуаций и условностей, тем самым еще являясь более гибким, в-третьих, использует сложные, каскадные и периодические запросы наряду с усложненной логикой их обработки.

Оба варианта частично являются самовольной трактовкой правил UML для более очевидного, но менее лаконичного и правильного описания проектируемых областей. В целом, однако, они следуют стандартам. Оба варианта подразумевают границу API, проходящую между процессами Application уровня (разрабатываемого приложения) и Social Network уровня (системы агрегации) и, опять же, деление на две области программы: система-агрегатор и агрегируемые системы. Также эти диаграммы отображают расширенную версию варианта использования для отражения гибкости приложения в контексте доступного пользователю функционала и функционала, образующего фундамент для перспектив.

Описание вариантов использования 1.1 и 2.1 должно сыграть положительную роль в проектировании и дальнейшей разработке, так как заключают в себе критический для работы функционал и являются частью критического пути использования системы.

Начертание и проектирование UML-диаграмм выполнялось согласно спецификациям UML от OMG.org[\[39\]](#), все условности оговорены.

Интеграция социальных сетей имеет отдельную сложность проектирования и разработки. Социальные сети в большинстве случаев имеют специальные, уникальные механизмы аутентификации и регистрации. По этой причине доступ через API к модели и методам той или иной соцсети осложнен дополнительными факторами[\[12\]](#).

Большая часть социальных сетей используют (иногда обязательно) двухфакторную авторизацию. Некоторые требуют подтверждения авторизации и так далее. В таком случае данная система может столкнуться с одной или более из этих особенностей. Для нормального отображения и прохождения вышеупомянутых процессов адаптируем проектирование.

В данной системе также будет присутствовать многоуровневая система приведения и проекции данных. Полученные данные из социальных сетей как минимум один раз будут преобразовываться и нормализовываться к виду, который будет удобен и эффективен для использования внутри системы. Это единственный работающий и адекватный способ отображать данные из разных социальных сетей, API которых возвращают данные в разном виде.

Несмотря на наличие СУБД в проектировании, она будет опущена как рекомендуемая часть при дальнейшем развитии проекта и потенциального вывода его в массы. В том же времени по причине достижения цели разработки и проектирования и недостаточной целесообразности внедрения такой технологии на данном этапе развития реализуемого ПО.

Отдельный скрининг и листинг процесса тестирования и дебаггинга будет опущен как рекомендуемая часть при дальнейшем развитии проекта и потенциального вывода его в массы. По той причине, что данный проект подразумевает конкретно анализ, проектирование, разработку, по причине недостаточного веса в противопоставлении другим главам разрабатываемой системы. Тестирование так или иначе будет проведено, но конкретно вывод и описания этого теста будут отсутствовать в тексте данной работе по уже упомянутой причине.

В целом этап фундаментального проектирования решения можно считать завершенным, большая часть результатов указана в соответствующих приложениях и пунктах разбора проектирования. Малая доля неоговоренных результатов применялись в ходе разработки и/или оговаривались отдельно в разных главах работы.

2.2 Используемые методологии и алгоритмы

Выбор методологий и алгоритмов — очень важный этап проектирования ПО. Конкретный набор методологий и алгоритмов должен поддерживать структуру системы, улучшая ее сильные места, а иногда и позволяя упрощать реализацию и добавлять новый функционал. В целом данный проект не обладает таким масштабом для подробного описания и взвешивания эффективности каждого из методов, это в то же время не отрицает того факта, что, имея выбор методологий, необходимо аргументировать выбор и принимать решения в сторону аргументации для сохранения последовательности разработки и проектирования.

Укажем набор ключевых методов и алгоритмов в разрабатываемой системе, поверхностно объяснив значимость и аргументируя выбор.

ООП — объектно-ориентированное программирование. Парадигма выбрана для данной системы из-за навыков реализации, логичного представления данных в виде объектов (сообщение, контакт и т. д.), а также по причине парадигмальной совместимости с агрегируемой системой и библиотеками.

Сложность структуры, многообразие и уникальность API не позволяет использовать принцип DRY («не повторяйся») и частично не позволяет использовать SOLID. По той причине, что так или иначе код будет реализовываться несколько раз по причине невозможности его обособления в отдельные функции. Обособление в отдельные функции идет в оппозицию SOLID. Нарушаются расширенные принципы ООП тем, что API не предоставляют функции одного образца, и все так или иначе принимают/возвращают разные значения. В ходе разработки были предприняты попытки по исправлению данного изъяна.

В то же время принципы KISS («проще — лучше») и YAGNI («тебе это не понадобится») были реализованы, так как отражают минималистичный критический подход к разработке ПО, которому следует и данная система.

Несмотря на это, система предлагает фундамент для расширения функционала и самой платформы благодаря реализуемым библиотекам.

Асинхронность — это очень важный элемент разрабатываемой системы. Система работает с разными API разных социальных платформ, выгружает разную информацию: сообщения, друзья, контакты и так далее, наличие асинхронности у методов — критическая необходимость. Скорость API сильно разнится, обработки ответов тоже из-за неоднородности, в это же время необходимо обеспечить плавность использования GUI для пользователя и общую работоспособность системы.

GUI — важный элемент разрабатываемого приложения. В первую очередь приложение создается для обычных людей, пользователей. Для высокого уровня QoL, UX необходимо реализовать качественный, удобный, понятный интерфейс для взаимодействия пользователя с системой. Разметка GUI будет выполнена с помощью функционала используемого фреймворка.

JSON и конфигурация — для сохранения пользовательской конфигурации и настроек будем использовать локально размещенные файлы в формате JSON. Будем хранить данные в соответствующих файлах.

Для отображения работы системы-приложения будем компилировать сборку в исполняемый файл с помощью фреймворка. В данном случае программа будет компилироваться для Linux (в DLL-файл) и для Windows (в EXE-файл). Исполняемый файл, во-первых, не предоставляет исходный код, что хорошо в рамках данной работы, так как лимитирует владельцу доступ к оригинальному написанному коду, во-вторых, обладает некоторой долей независимости, то есть при некоторых условиях может быть запущен где бы то ни было, самостоятельно от зависимостей и ОС.

В ходе разработки и в процессе работы программы система будет взаимодействовать с черными ящиками API социальных сетей. В ходе работы будут обработаны две социальные сети. «Telegram», «VK». В таком случае, соответственно, будут использованы следующие технологии для сверки и разработки/реализации (порядок от агрегируемого ПО к разрабатываемому).

«Telegram»:

1. Telegram API (<https://core.telegram.org/>). Черный ящик;
2. TDLib[38]. Модели: TL.Client, TL.User, TL.Chat, TL.InputPeer, TL.Message, TL.MessageBase, TL.MessageService;
3. WTelegram[41], далее WT для сокращения объема. Модели: WT.Client, WT.Messages_GetAllDialogs, WT.UserOrChat, WT.Messages_GetHistory, WT.LoginUserIfNeeded, WT.GetAllChats, WT.SendMessageAsync.

«VK»:

1. VK Standalone Application API. Черный ящик;
2. VK API[34]. Модели не инстанцируются в ходе работы;
3. VkNet[40], далее VKN для сокращения. Модели: VKN.Client, VKN.Authorize, VkNet.Friends.GetAsync, VKN.FriendsGetParams, VKN.Messages.GetHistory, VKN.Messages.GetMessage, VKN.Client.Send.

В таком случае необходимо будет работать с документациями, методами, моделью этих API, доверяя владельцу соцсети в наличии высокого качества и высокой скорости предлагаемой услуги, то есть услуги наличия и доступа к API. Этот конкретный аспект оговорен в недостатках системы.

Также стоит отметить зависимость от библиотек, фреймворка, ЯП, ОС и прочих элементов воспроизведения работы реализуемой системы. В любом случае разрабатываемое ПО также несет ответственность за встраиваемое и используемое ПО третьих лиц перед клиентом, будучи посредником и провайдером той или иной услуги.

2.3 Требования к системе

В таблице 1 указаны минимальные требования к системе. Минимальные параметры в данном случае — это параметры, на которых эта система была запущена и протестирована. Также добавлена пометка о точности каждого конкретного требования в контексте разрабатываемой системы и возможные

расширения из-за зависимостей и в ходе тестирования. Эта пометка выделена курсивом.

Таблица 1 — Технические требования для использования разрабатываемой системы на двух разных семействах ОС.

	Windows	Linux
Операционная система (ОС) [43]	Windows 10, 22H2, 64х, сборка 19045. <i>От требования .NET 8.0 лучше чем: Windows 10, сборка 1607</i>	Ubuntu 22.04, Mint 21.3 x64. <i>От требования .NET 8.0 лучше чем: Ubuntu 20.04, Debian 11, Fedora 38</i>
Графическое процессорное устройство (ГПУ)	Nvidia GTX 750Ti, 2048 Мб. <i>В ходе работы системы почти не используется.</i>	Intel Jasper Lake UHD, память делиться с системой. <i>Минимальный порог для работы приложения</i>
Центральное процессорное устройство (ЦПУ)	Intel Core i7-4790 4 x 3.60 ГГц. <i>Средняя нагрузка при работе менее 5%</i>	Intel Celeron N4500 2 x 1.10 ГГц. <i>Средняя нагрузка при работе менее 15%</i>
Оперативное запоминающее устройство (ОЗУ)	7.8 Гб. <i>В ходе работы программы использовалось около 100-150 Мб</i>	3.6 Гб. <i>В ходе работы программы использовалось около 100-120 Мб</i>
Постоянное запоминающее устройство (ПЗУ)	256 Гб (твердотельный диск). <i>В конечном итоге занимает 5 Мб</i>	128 Гб (жесткий магнитный диск). <i>В конечном итоге занимает 2 Мб</i>

Помимо технических требований к системе, она также должна выполнять ряд требований другого характера в свете многочисленных причин, привязанных к разрабатываемой/проектируемой системе, объекту, сфере, методологии, технологии применения.

Требования к безопасности связаны с работой системы с персональными данными. В том числе паролями, телефонами, именами, а также в случае с двухфакторной авторизацией — с вторичными факторами личной информации. От системы требуется следовать нормам безопасности работы с персональными и общими данными. Требуется полная изоляция процесса доступа к API. Серьезный изъян, связанный с безопасностью системы,

отмечен в разделе недостатков. При хранении данных на устройстве требуется выполнять полный цикл шифрования данных, включая кодирование, соль, хэширование. Стоит учитывать законодательство страны и регулировки, связанные с персональными данными, в условиях разработки в РФ эти регулировки содержатся в источниках[\[1\]\[2\]\[3\]](#), этот список фундаментален, но недостаточен, следует проконсультироваться с юристами.

Для получения необходимых данных с серверов социальных сетей необходимо использовать предоставленные API (желательных первых лиц), разработанные сторонними технологии следует проверить технически и юридически, а также на их защиту данных.

Дизайн такой системы должен быть удобным и интуитивно понятным. Система в первую очередь разрабатывается для людей, где ключевая аудитория заключена обычными людьми разного технического образования. В таком случае помимо руководства пользователя необходимо создать интерфейс максимально доступным и элементарным для пользования системой любым клиентом.

Все функции и элементы управления следует делать доступными и простыми в использовании. Текстовая информация и элементы интерфейса должны быть четкими и легко читаемыми. Необходимо выбрать шрифты и размеры текста, которые обеспечат удобное восприятие информации. Интерфейс приложения должен быть простым и минималистичным. Оно должно быть бесшовным, чтобы опыт пользователя максимально отражал цель проекта — простое общение без границ социальных сетей. Должно быть как можно меньше намеков на то, что платформа — это агрегатор.

Необходимо выполнить требования к разнообразию операционных систем и технических требований. Чем ниже требования, тем больше аудитория, чем больше поддерживаемых ОС, тем больше аудитория. Следуя этому, необходимо обеспечить достаточную совместимость с разными операционными системами, в первую очередь включая Windows, Linux.

3 РАЗРАБОТКА

3.1 Общие сведения системы

Проектируемая система представляет собой десктопное приложение-мессенджер для агрегирования содержания различных социальных сетей, которое позволяет принимать сообщения из разных социальных сетей и отправлять их. Такая система должна как минимум позволять пользователю общаться с другим пользователем, при этом игнорируя саму платформу общения.

Приложение будет реализовано для Windows, а также потенциально для UNIX/Linux платформ, успешный тест на Ubuntu без UI. Оно будет иметь вид десктопного приложения с интерфейсом взаимодействия пользователя с системой. Приложение будет требовать доступа к интернету для работы. Оно будет постоянно использовать сторонние инструменты для достижения цели и работать с API (нативными и третьих сторон) непрерывно. Пользователь должен будет авторизоваться, используя свои личные данные, на каждой из социальных сетей, которые он хочет подключить. Пользователь будет иметь GUI с определенным уровнем качества UX/UI для взаимодействия с элементами системы. Система может быть развернута бесплатно через GitHub в разделе release, через любой другой хостинг приложений и контента.

В разделе 3.2 указан набор технологий, использованных в ходе разработки. Раздел 3.3 описывает поэтапно ход разработки. Раздел 3.4 предоставляет результат разработки. В разделе 3.5 описаны перспективы. Раздел 3.6 описывает актуальные и потенциальные, критические, серьезные и незначительные недостатки.

Плановые сроки начала и окончания работ по созданию системы: с 03.05.2024 по 15.06.2024. Развертка и публикация потенциально она может быть выполнена, но не будет описана в рамках данной ВКР.

3.2 Стек разработки и используемые технологии

Выбор технологий — очень важный этап разработки, проектирования и внедрения программного обеспечения. Правильный стек технологий позволяет выполнять работу качественно, быстро и эффективно, помогает избегать сложностей и в целом составляет большую долю успеха при разработке и проектировании той или иной системы.

Выбор каждого из элементов стека основан на множестве факторов, начиная с наличия опыта работы, заканчивая стоимостью и фактором направления. В таком случае следует сделать правильный выбор и аргументировать этот выбор в процессе отбора.

Учитывая, что данная работа заключена не только в разработке, но и в проектировании и анализе, данный стек будет покрывать весь спектр технологий. Следовательно, не все из перечисленных технологий так или иначе применялись конкретно для разработки системы. Многие использовались лишь косвенно, например для упрощения создания документации. Наличие в этом списке сторонних, на первый взгляд избыточных упоминаний — следствие аргументации, приведенной ранее.

В таблице 2 помимо упоминания реализуемой технологии присутствует краткое пояснение к спектру и месту применения и аргументация к выбору данной технологии из всех аналогов.

Таблица 2 — полный стек разработки и проектирования, порядок по возрастанию.

Технология	Пояснение и аргументация
C# 12.0	<i>ЯП.</i> C# был выбран в связи с: наличием асинхронности для множественных запросов и непоследовательной их обработкой, LINQ для фильтрации и выборки из множества с помощью внутренних SQL-подобных запросов, ООП для структуризации и объектного представления, наличием библиотек для достижения цели, наличии форумов и поддержки для помощи с разработкой, а также с его современностью. В тоже время с наличием опыта работы и open-source сред разработки.
WTelegram 4.1.1	<i>API third-party библиотека для запросов/ответов.</i> Сторонняя библиотека программируемого интерфейса приложения Telegram. Выбрана из-за лаконичности, простоты использования, наличия минимального необходимого функционала для достижения цели. Является open-source проектом. В разделе недостатков системы упомянут достаточно серьезный изъян выбранной технологии. Причина выбора WTelegram над нативным TDLib Telegram также оговорена в недостатках системы.
VkNet 1.78.0	<i>API third-party библиотека для запросов/ответов.</i> Сторонняя библиотека для программируемого интерфейса приложения VK. Выбрана из-за минималистичности, элементарности использования, наличия критического функционала для достижения цели. Является open-source проектом. В разделе недостатков системы упомянут достаточно серьезный изъян выбранной технологии. Причина выбора VkNet над нативным API VK также оговорена в недостатках системы.
Nuget 6.10.0	<i>Система управления библиотеками и пакетами.</i> Альтернатив с такой большой базой инструментов нет.
.NET 8.0	<i>Фреймворк.</i> .NET был выбран в связи с JIT и сборщиком мусора, CIL и возможностью использовать функциональный F# в ряде случаев, наличием большой базы документации и поддержки, наличием кроссплатформенности (в отличие от .NET Framework), длительным сроком поддержки (до 2026), в то же время поддержкой Nuget и компиляцией в объектный и исполняемый файлы.
.NET MAUI 8.0	<i>Фреймворк.</i> Имеет ту же специфику что и .NET, но нацелен на разработку для Linux/iOS/Android. Был использован для создания консольного приложения для тестирования работы.
Visual Studio 2022 17.10.1	<i>IDE.</i> Нативная среда разработки для приложений на .NET и C#. Выбрана из-за опыта использования, наличия GUI для дизайна WPF/.NET. Наличием поддержки работы с Git, наличием дебагера с отслеживанием.

Продолжение таблицы 2

JetBrains dotPeek 2024.1.2	<i>Инструмент.</i> Для декомпиляции C#. Применен для восстановления потерянной части кода.
TDLib	<i>Native API.</i> Интерфейс взаимодействия для приложений, работающими с Telegram.
VK API	<i>Native API.</i> Интерфейс взаимодействия для приложений, работающими с VK.
Telegram API	<i>Native API.</i> Черный ящик Telegram. Принципы получения запросов и ответа на них.
VK Standalone App	<i>Native API.</i> Черный ящик VK. Принципы получения запросов и ответа на них.
Draw.io	<i>Графический инструмент.</i> Был выбран по причинам: open-source, гибкость, многофункциональность.
GIMP 2.10.32	<i>Графический инструмент.</i> Использован для стилизации отдельных элементов UX/UI. Бесплатный, многофункциональный аналог Photoshop.
Google Docs	<i>Редактор текста.</i> Бесплатный, онлайн-версия, гибкий в плане работы со ссылками, таблицами. Применен в ходе написания спецификации.
GitHub	<i>Децентрализованная система управления версиями.</i> Наиболее популярный, функциональный, доступный вариант VCS. Выбран из-за наличия опыта работы, доступности, удобства.

3.3 Этапы разработки

Для отображения процесса разработки и для подтверждения следования этапам, указанным в ходе проектирования и постановки задач, будем вести документацию этапов разработки. Этапы имеют исчерпывающий характер и покрывают весь процесс разработки, исключая лишь подробность описания. При достижении последнего этапа разработки должно быть разработано работающее приложение, все задачи уровня разработки должны быть выполнены. Условности в описании некоторых этапов связаны с: упрощением понимания с точки зрения задумки, с сокращением объема описания, с отсутствием необходимости полноценного описания этапа (очевидные, само собой разумеющиеся аспекты).

Этапы разработки коррелированы со стеком, но не напрямую, разные технологии могут быть использованы на разных этапах. Все технологии так или иначе присутствуют в этапах разработки.

Ход работы представлен следующими этапами:

1. **Организация рабочей среды.** Первый этап разработки. Создание приемлемых условий для разработки. Организация работы, среды, файлов, директорий;
 - 1.1. Создания проекта в IDE. Создаем проект по шаблону Windows Forms Application в Visual Studio 2022. Выбираем фреймворк .NET версии 8.0, название проекта и его расположение;
 - 1.2. Установка требуемых пакетов и библиотек. Через Nuget устанавливаем необходимые пакеты либо с помощью GUI, либо через Packet Manager Command Line. Главным образом устанавливаем WTelegram, VkNet и их зависимости;
 - 1.3. Организация работы с GitHub. Инициализируем локальный репозиторий в директории. Создаем приватный удаленный репозиторий, настраиваем gitattributes, gitignore, отталкиваясь от разрабатываемой системы. Пушим .lock для завершения инициализации без файлов;
 - 1.4. Создаем необходимые файлы, организуем директорию. Директория и расположение файлов будут иметь вид, указанный в самом левом столбце [\[Приложения Д\]](#);
 - 1.4.1. Properties включает в себя resx используемых ресурсов, и два datasource для отображения сообщений и контактов в DataGridView;
 - 1.4.2. Configs включает в себя json файлы разных разделов конфигурации, а также классы для отображения этих данных внутри модели;

- 1.4.3. Forms включает в себя файлы-исходники для отображения разных окон наряду с реализациями некоторых методов этих окон;
- 1.4.4. Resources включает в себя статичные файлы, изображения, иконки;
- 1.4.5. Sources включает в себя файлы-исходники для модели, методов главной формы, функций, перегрузок. В целом вся логика приложения находится в этой директории;
- 1.5. Подготовим файл документации для параллельного документирования разработки;
- 2. **Реализация уровня модели.** Проектируем большую часть классов и интерфейсов из UML-диаграммы классов в кодовую реализацию. Ставим заглушки для методов. Отметим, что конечная реализация может отличаться от проектирования;
 - 2.1. Создаем интерфейсы: ISocialNetwork, IContact, IMessage;
 - 2.2. Создаем классы: VKSocialNetwork, TelegramSocialNetwork, OkSocialNetwork, ViSocialNetwork (имплементации интерфейса ISocialNetwork). AnyContactType (имплементация IContact). AnyMessageType (имплементация IMessage);
 - 2.3. Создаем все forms для всех требуемых окон приложения;
 - 2.4. Загружаем и организуем все статичные файлы, ресурсы, создаем binding data sources, которые должны ссылаться на внутренние типы. Один data source ссылается на список IContact, второй — на список IMessage;
 - 2.5. Объявляем все методы/функции/поля каждого из типов, шаг может быть выполнен вне хода работы по причине ошибок проектирования. В конечном итоге набор методов/функций/полей представлен в [\[Приложении Д\]](#) в правой его и центральной части;
- 3. **Реализация критического функционала.** Пишем код для критического функционала, в первую очередь разбираемся с вариантом использования

1.1, затем с 2.1. Прописываем поведения внутри уже созданной модели. Идем от меньшего к большему;

- 3.1. Пишем методы для ISocialNetwork, IContact, IMessage. Ключевая выборка описания методов/полей/функций в [\[Приложении Е\]](#) (описание некоторых методов убрано);
- 3.2. На данном этапе прописываем использование сторонних методов и моделей, API третьих лиц и нативного API социальных платформ;
- 3.3. В конечном итоге мы имеем каркас работающей системы, алгоритмы которого способны отдельно друг от друга выполнять весь требуемый функционал. То есть у нас есть методы, модели, модули для будущего объединения в единую работающую систему.

4. **Реализация уровня контроллера.** Пишем код для методов контроля модели и преобразования данных. В данном случае подразумевается нормализация данных, полученных в разделе 3, для их нормального отображения на уровне вида;

- 4.1. Пишем методы внутри Form1.cs (конструктор, список полей, методы инициализации), EventHandlers.cs (список методов для обработки событий при взаимодействии с пользователем с GUI), Functions.cs (функции для обработки вида, обособления функций из методов событий), Overrides.cs (перегруженные нативные методы WinForms). Ключевая выборка в [\[Приложении Ж\]](#);
- 4.2. В конечном итоге при выполнении данного этапа происходит связывание и объединение всех подсистем и всех методов и алгоритмов в единое целое;
- 4.3. На данном этапе разработки у нас выполнена вся работа с алгоритмами, технологиями и методологиями для получения данных, с обработкой данных и их нормализацией, с проектированием данных в разрабатываемую систему. Также присутствуют методы для обратного процесса приведения

введенных в систему данных к виду данных внутри утилизируемых социальных сетей;

5. **Реализация уровня вида.** Создаем визуальное представление приложения, которое при этом в первую очередь способно отображать данные, которые приложение получает после этапа уровня контроллера. Используем GUI Visual Studio для создания дизайна и UI/UX.

3.4 Работа программы и интерфейс

На рисунке 1 изображено главное меню программы. Имена, текст сообщений и подобные элементы закрыты для формальности процесса документации и для сохранения тайны переписки. В левой части — список контактов, цветом помечен источник (социальная сеть). В правой части — логотип и статистика приложения. Сверху в заголовке слева направо: логотип, название, кнопка настроек, кнопка руководства пользователя, кнопка об авторе, кнопка свернуть, кнопка на весь экран, кнопка закрыть. Во втором ряду: кнопка «Контакты» сбрасывает выбранный контакт, поле «Поиск» — фильтрация контактов по имени. При нажатии на контакт из списка слева в правой части открывается беседа.

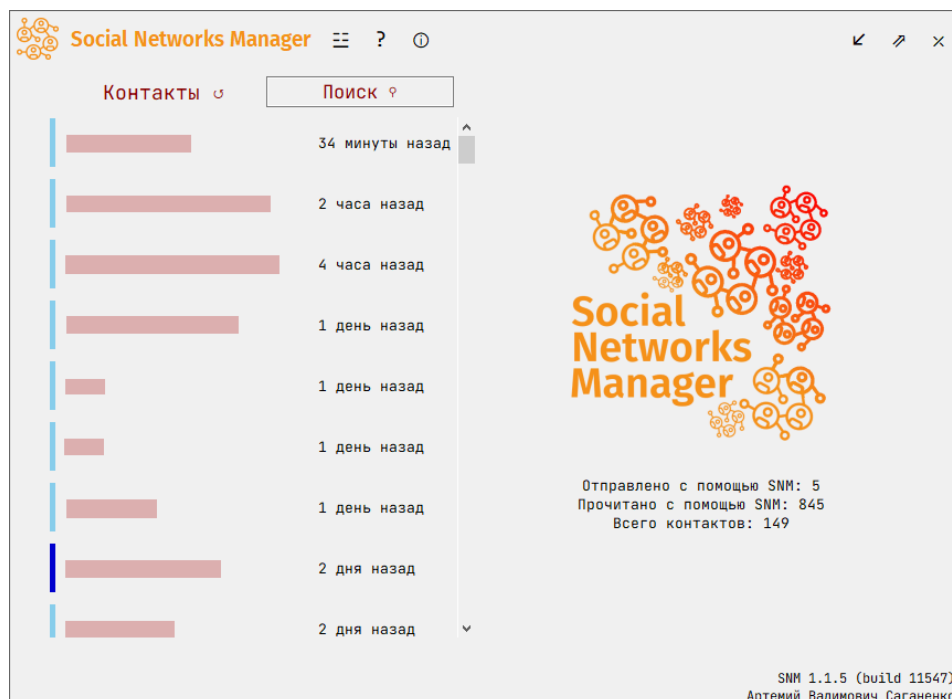


Рисунок 1 – Главное меню

На рисунке 2 изображен главный экран при открытой беседе. В левой части — список контактов, синим (самый первый) помечен выбранный контакт, беседа которого отображается. Справа — часть с беседой. Во втором ряду: кнопка «Сообщения» — для обновления списка сообщений и возврата к последнему сообщению, кнопка «Открыть» — для открытия контакта в браузере для прямого общения. Далее в правой части — список сообщений. Слева направо: если оранжевая метка слева — сообщение входящее, справа — исходящее, далее — метка времени, если сообщение входящее — время сообщения слева, исходящее — справа, затем идет текст сообщения (посередине). Внизу правой части находятся поля для ввода текста сообщения, и правее кнопка — для отправки сообщения.

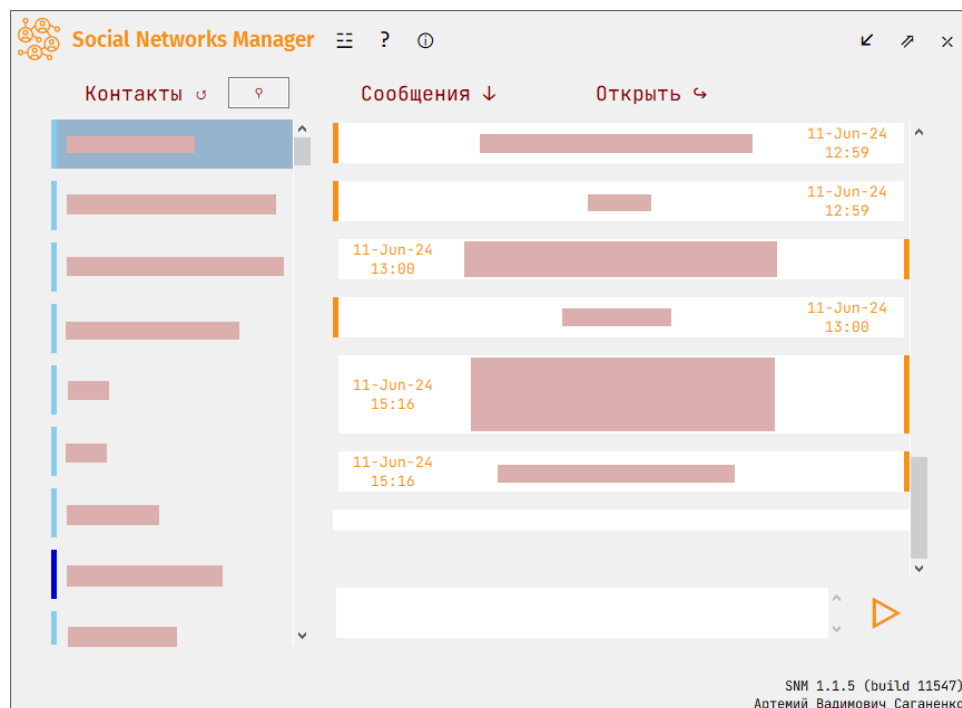


Рисунок 2 – Главное меню, контакт активирован, справа область беседы

На рисунке 3 изображено окно настроек. Делим окно на условные пять частей: слева сверху, справа сверху, слева снизу (с красным текстом), справа снизу (с красным текстом). Рассмотрим их в порядке перечисления. Первая часть с настройками внешнего вида и с настройками фильтрации, три кнопки в первом ряду отвечают за визуальное представление системы, реализация на данный момент отсутствует, во втором ряду кнопки и счетчик для правил

фильтрации для сокрытия контактов, сообщений, реализации на данный момент нет. Вторая часть с настройками агрегации, две колонки, изображения агрегируемых систем и кнопки для перехода к подключениям, реализовано. Третья и четвертая часть сделана для открытия соответствующих файлов конфигурации, по мере прогресса реализации и дополнения функционала будут убраны. Пятая часть подразумевает две кнопки для сохранения изменений или их сброса.

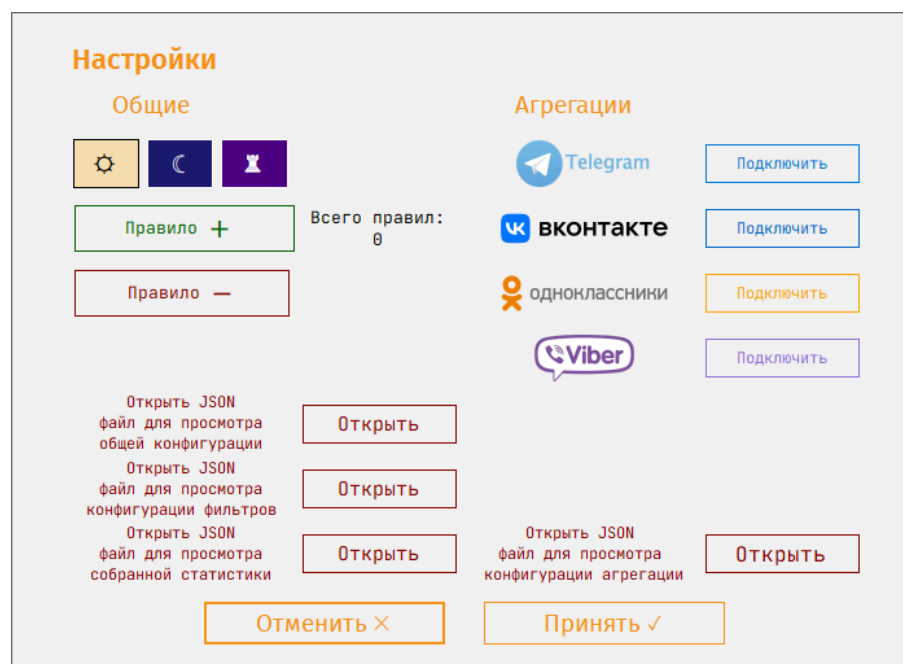
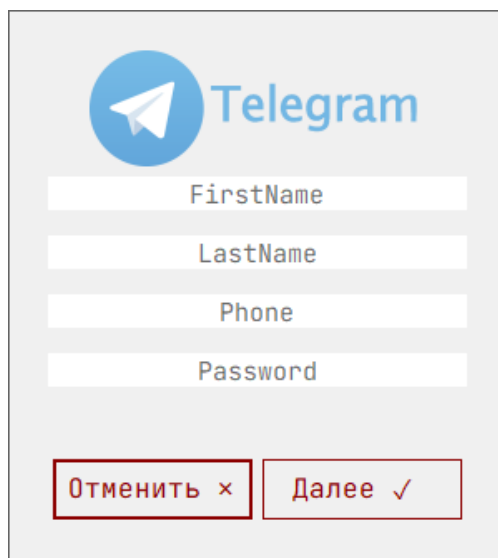


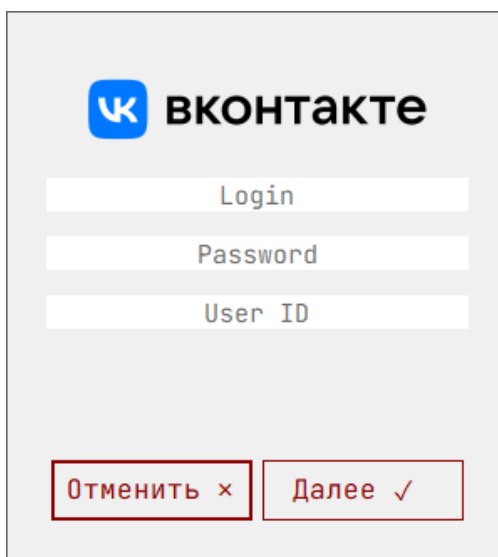
Рисунок 3 – Окно настроек

Рисунки 4, 5, 6 отображают процесс подключения агрегации. Рисунок 4 — окно агрегации для «Телеграма», рисунок 5 — окно агрегации для «ВК», рисунок 6 — окно двухфакторной авторизации. В окнах авторизации агрегации необходимо заполнить все поля и нажать далее для перехода к двухфакторной авторизации (если нужно). В случае нажатия отмены авторизация сбрасывается. В окне двухфакторной авторизации необходимо набрать код и нажать далее.



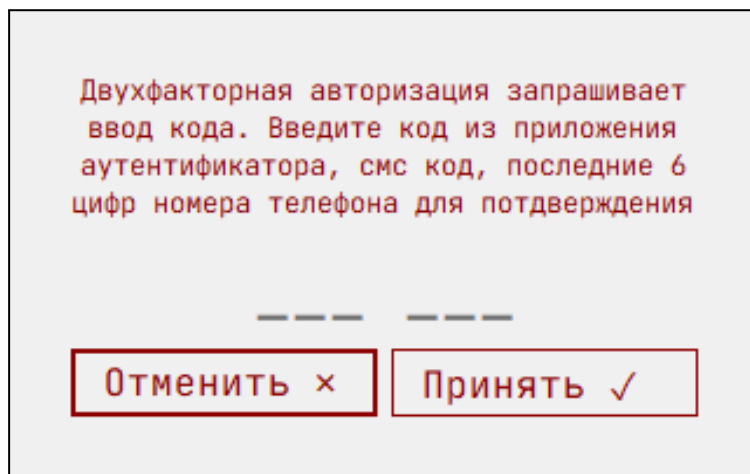
The image shows a Telegram authorization form. At the top is the Telegram logo (a blue circle with a white paper plane) and the word "Telegram" in blue. Below the logo are four input fields: "FirstName", "LastName", "Phone", and "Password". At the bottom are two buttons: "Отменить ×" (Cancel) and "Далее ✓" (Next).

Рисунок 4 – Авторизация «Телеграма»



The image shows a VK authorization form. At the top is the VK logo (a blue square with a white "VK" icon) and the word "ВКонтакте" in black. Below the logo are three input fields: "Login", "Password", and "User ID". At the bottom are two buttons: "Отменить ×" (Cancel) and "Далее ✓" (Next).

Рисунок 5 – Авторизация «ВК»



The image shows a two-factor authentication window. The text inside reads: "Двухфакторная авторизация запрашивает ввод кода. Введите код из приложения аутентификатора, смс код, последние 6 цифр номера телефона для подтверждения". Below the text is a dashed line representing the code input field. At the bottom are two buttons: "Отменить ×" (Cancel) and "Принять ✓" (Accept).

Рисунок 6 – Окно «Двухфакторная авторизация»

На рисунке 7 изображено окно «Об авторе» для отображения создателя проекта, названия проекта и руководителя.



Рисунок 7 – Окно «Об авторе»

3.5 Перспективы развития и нереализованный функционал

В будущем разрабатываемую систему можно развивать и улучшать в зависимости от требований пользователей, научно-технического прогресса, а также с улучшением API, используемых библиотек, навыков разработчиков.

Вследствие специфики разрабатываемой системы необходимо обеспечить большую гибкость и масштабируемость системы. Следует учитывать не только внутренние требования к данной системе, но и ограничения, введенные внешними условиями. Эти ограничения имеют гораздо более сильный эффект в свете того, что разрабатываемая система является лишь десктопным приложением, а запросы от разрабатываемой к сторонним агрегируемым системам через API могут со временем стать слишком частыми и объемными, а их количество может расти экспоненциально. Это необходимо учитывать при улучшении системы.

Далее опишем потенциальный функционал в порядке от самой ближней перспективы к самой дальней. Не будем углубляться в детали. Поверхностно опишем методологию и технологию, сложность разработки.

1. Отображение изображений профилей пользователей (аватаров), изображений чатов. Отображение последнего времени в сети, последнего сообщения контакта. Доступные методы в: WTelegram, VkNet. Малая сложность имплементации через DataGridView контактов (объект имеет тип колонки «изображение», «текст»);
2. Добавление настроек внешнего вида системы и фильтров сообщений и контактов. Имеется навык реализации, малая сложность;
3. Работа с приложениями к сообщениям, любые медиафайлы. Есть специальные методы в WTelegram, VkNet. Реализация через внутренние объекты. Очень высокая сложность;
4. Добавление отображения групп из «Телеграмм». Есть специальные методы в WTelegram. Средняя сложность;
5. Локализации на другие языки. Есть навыки реализации. Малая сложность;
6. Объединение контактов в один для представления одного человека, который имеет аккаунты в разных социальных сетях. Средняя сложность;
7. Полная кросс-платформа системы. .NET поддерживает кросс-платформу. Слабые навыки реализации. Средняя сложность. Следует изучить и использовать .NET MAUI;
8. (Условное) Улучшение методов авторизации и агрегации. Переход к другим библиотекам или нативному API. Нет навыков. Крайне высокая сложность, см. недостатки системы;
9. (Условное) Консультация, договор с разработчиками «ВК» для устранения запрета на сбор сообщений с помощью VkNet, см. недостатки системы;

10. Монетизация через встраиваемую рекламу и премиум-подписки. Работа с еще одним набором API. Очень высокая сложность;
11. Добавление агрегации других социальных сетей, включая иностранные, прочих мессенджеров, прочих сетей. Неизвестная сложность;
12. Добавление функционала добавления контактов через десктопное приложение WTelegram, VkNet имеет методы для реализации. Средняя сложность;
13. Улучшение, добавление нового функционала для идеи карты пользователей, где каждый человек может общаться с другим ближайшим географически человеком, который использует эту систему. Наподобие системы [\[30\]](#).

В целом пользователь может полностью заменить использование социальных сетей с помощью данной платформы, при этом сокращая время проведения в социальных сетях и упрощая опыт своего общения с людьми вокруг. При этом в случае качественной реализации данный проект может иметь финансовую выгоду в свете внедрения рекламы и монетизации.

3.6 Недостатки системы

Следует отметить, что любая система имеет недостатки, это неизбежно для проекта любого масштаба, который разрабатывает разное количество людей или один человек, для команд разной компетенции, профессионализма и даже характера. В любом случае недостатки необходимо в лучшем случае исправлять, в худшем — скрывать от клиента, человека или компании, которые пользуются этой системой. Последний вариант больше подходит для реальных бизнес-проектов, которые зарабатывают деньги. Опять же, это неправильно, но в бизнесе, к сожалению, нет места для репутационных и экономических потерь. Первый вариант больше подходит для технических, научных, исследовательских и подобных систем, где общая правильность и работоспособность имеет критический эффект.

Это в большой степени относится к проектам в рамках MVP, PoC, экспериментальным и нестабильным решениям, а также к разработкам в сфере цифровых коммуникаций, социальных сетей, систем, взаимодействующих с людьми, и тому подобное. Это следствия выполненного на скорую руку проекта либо проекта, работающего с абстрактными неисчислимыми сущностями.

Еще один аспект недостатков данной системы — это юридические вопросы относительно работы с персональными данными и со сторонними/нативными API разных систем. Также стоит учитывать изъяны, связанные с выбранными библиотеками для реализации функционала. Полный список недостатков представлен далее.

- Юридические вопросы, связанные с законодательством Российской Федерации, в первую очередь с федеральным законом «О персональных данных»[\[3\]](#), федеральным законом «Об информации, информационных технологиях и о защите информации»[\[2\]](#), статьями 23 и 24 Конституции РФ[\[1\]](#). Стоит обязательно учитывать способ хранения данных, их шифрования, передачи и других аспектов. Несмотря на то, что данная система напрямую не является соцсетью, она как их агрегатор перенимает часть юридических требований. Проблема может быть решена при дальнейшей разработке, при условии консультации с юристами;
- Техническая проблематика, связанная с запретом «VK» на сбор сообщений через API без предварительного договора с руководством[\[18\]](#). Проблема лишает разрабатываемую систему большей части смысла, но может быть решена;
- WTelegram[\[41\]](#), VkNet[\[40\]](#) и прочие сторонние библиотеки в конечном итоге должны быть лишь временной заглушкой, со временем необходимо прийти к альтернативе. Работа с нативным API гораздо более надежна в плане защиты информации, сохранения функционала во времени с развитием систем, проще в законодательном плане, имеет

больший функционал и в целом является более правильной методологией доступа к backend API той или иной платформы. Разработка такого аспекта крайне сложная, но может быть выполнена в длительной перспективе;

- В любом случае работа со сторонними ПО/API, даже нативными, несет определенные сложности и недостатки. По сути, используя стороннее ПО/API, мы перенимаем ответственность за недостатки и проблемы стороннего ПО/API на себя, что иногда может быть крайне невыгодно как для бизнеса, так и для разработки. Чрезмерная надежда на качество и исполняемость услуг, предложенных другими, определяет нашу систему как зависимую и неустойчивую. Нет решения проблемы, так или иначе, мы работаем с другими системами, это одна из черт данной системы;
- Небольшой просчет рынка и недостаток, связанный с переоценкой глубины и силы соцсетей на рынке в России. Не самый критический аспект в плане разработки ПО, но достаточно важный в контексте идеи разработки. Старое исследование [\[25\]](#) описывает слабость и монополию на рынке социальных сетей. Данная система делает упор на капиталистический свободный рынок, которым настоящий рынок соцсетей частично не является.

3.7 Руководство пользователя

Согласно этапам разработки, указанным в одноименном разделе, одним из элементов было создание руководства пользования. Очевидно, что разрабатываемая система создается в первую очередь для людей, пользователей социальными сетями. В указанном контексте подразумевается необходимость создания руководства для применения программы новыми клиентами в целях упрощения их опыта использования и в целом для повышения QoL и UX.

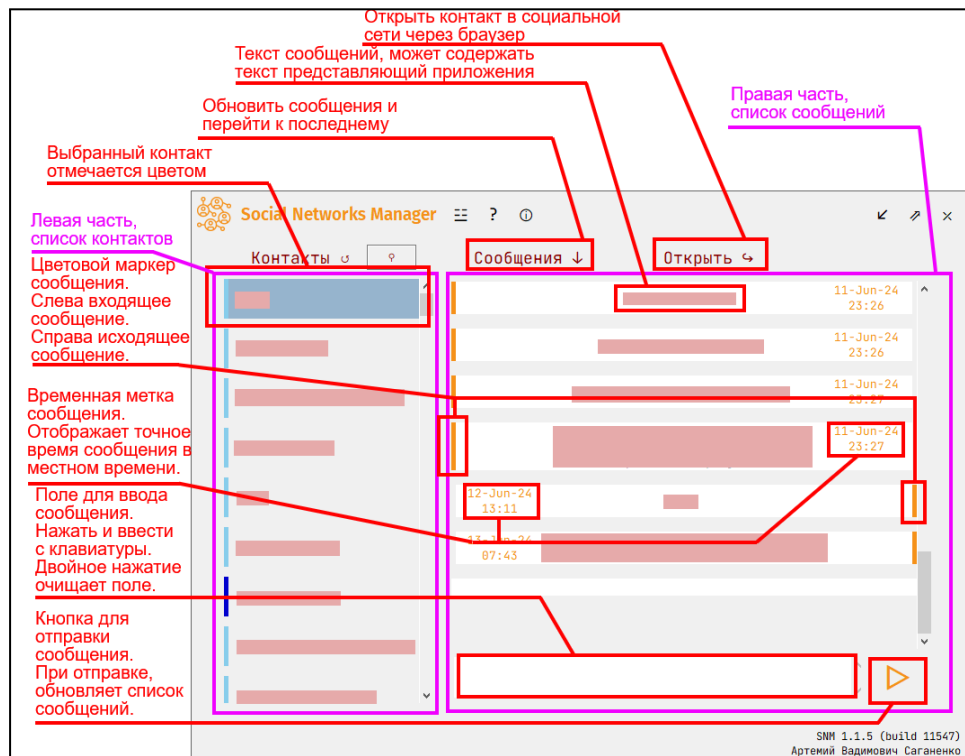


Рисунок 9 - Руководство пользователя, главное меню с открытой беседой

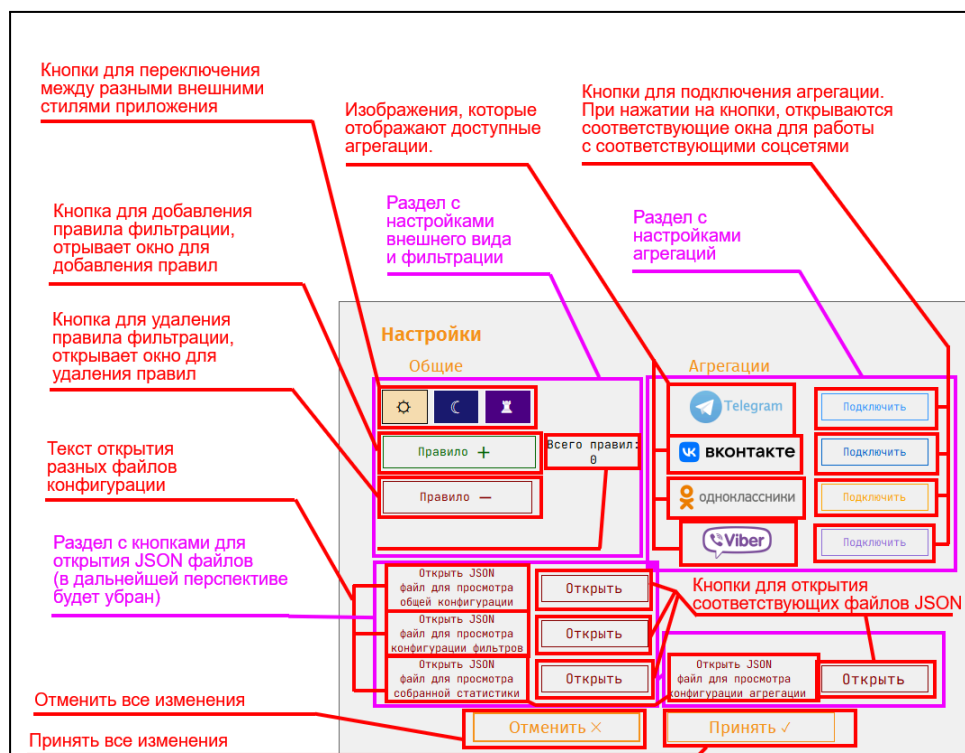


Рисунок 10 - Руководство пользователя, меню настроек приложения

ЗАКЛЮЧЕНИЕ

Целью данной работы были проектирование и последующая разработка программного обеспечения, нацеленного на агрегацию содержания социальных сетей пользователя. Система должна была в первую очередь предоставлять возможность пользователю отправлять сообщения в разные источники из одного приложения и получать входящие сообщения из разных источников.

С достижением успеха во всех поставленных задачах и реализации конечного ПО главную цель данной работы может считать достигнутой.

С точки зрения теоретической значимости и составляющей, в данной работе был проведен анализ разработки ПО в социальной сфере и в объекте социальных сетей и цифровых коммуникаций. Также был проведен отбор методологий и технологий, реализуемых в сфере. В итоге был выбран наиболее эффективный стек технологий и набор методологий для реализации поставленной цели с учетом всех аспектов. Все аспекты системы и все выборы, принятые в ходе анализа, проектирования, разработки, так или иначе аргументированы.

С точки зрения практической значимости и составляющей, в данной работе был проведен полный цикл проектирования и разработки системы. В конечном итоге было разработано прикладное ПО, которое в определенной степени выполняет поставленную прикладную цель. Степень выполнения, то есть степень завершенности ПО, устанавливается тем, что ПО по своей сути является MVP/РoC-разработкой. Оно в достаточной степени эффективно работает с поставленной проблематикой.

В конце работы были предоставлены: документация, код, реализация (сборка), инструкция пользователя, теоретическое обоснование, практическое обоснование, отчет, документ требований, документ системных требований, титульный лист, календарный план, лист согласований сведений, реферат. Исключение составило: тезис, концептуальный и дизайн-документы.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Конституция Российской Федерации : [принята всенародным голосованием 12 декабря 1993 г. с изменениями, одобренными в ходе общероссийского голосования 01 июля 2020 г.] // Официальный интернет-портал правовой информации. URL: <http://publication.pravo.gov.ru/Document/View/0001202007040001> (дата обращения: 23.05.2024).
2. Федеральный закон Российской Федерации от 08.07.2006 г. № 149-ФЗ (ред. от 25.11.2017) «Об информации, информационных технологиях и о защите информации» // Российская газета. – 2006. – № 165. URL: <https://rg.ru/documents/2006/07/29/informacia-dok.html> (дата обращения: 04.05.2024).
3. Федеральный закон Российской Федерации от 08.07.2006 г. № 152-ФЗ (ред. от 02.07.2021) «О персональных данных» // Российская газета. – 2006. – № 4131. URL: <https://rg.ru/documents/2006/07/29/personalninye-dannye-dok.html> (дата обращения: 09.05.2024).
4. Аудитория восьми крупнейших соцсетей в России в 2023 году: исследования и цифры // ppc.world URL: <https://ppc.world/articles/auditoriya-vosmi-krupneyshih-socsetey-v-rossii-issledovaniya-i-cifry/> (дата обращения: 02.05.2024)
5. Введение в анализ социальных сетей на примере VK API // Хабр URL: <https://habr.com/ru/articles/263313/> (дата обращения: 09.06.2024).
6. Воробьева И.А., Костерев Р.А. Цифровая коммуникация в структуре цифрового общества // Международный научно-исследовательский журнал. – 2022. – № 6-4 (120). URL: <https://cyberleninka.ru/article/n/tsifrovaya-kommunikatsiya-v-strukture-tsifrovogo-obschestva> (дата обращения: 09.05.2024).
7. В социальных сетях слабые связи важнее, чем сильные // Хабр URL: <https://habr.com/ru/articles/7494/> (дата обращения: 10.06.2024).
8. ГОСТ Р 7.0.100-2018. Библиографическая запись. Библиографическое описание. Общие требования и правила составления : национальный стандарт Российской Федерации : дата введения 2019-07-01 / Федеральное агентство по техническому регулированию. – Изд. официальное. – Москва : Стандартинформ, 2018. – 124 с.
9. Децентрализация социальных сетей // Хабр URL: <https://habr.com/ru/articles/167653/> (дата обращения: 24.05.2024).
10. Добринская Д.Е. Что такое цифровое общество? // Социология науки и технологий. – 2021. – № 2. URL: <https://cyberleninka.ru/article/n/cto-takoe-tsifrovoe-obschestvo> (дата обращения: 09.05.2024).
11. Идеальная социальная сеть // Хабр URL: <https://habr.com/ru/articles/123132/> (дата обращения: 25.05.2024).
12. Интеграция сайтов и социальных сетей: эргономические аспекты // Хабр URL: <https://habr.com/ru/companies/alee/articles/122617/> (дата обращения: 01.06.2024).
13. Исследование Mail.Ru Group: социальные сети в России // hi-tech.mail URL: <https://hi-tech.mail.ru/news/18206-social-mail/> (дата обращения: 03.05.2024).
14. Исследование соцсетей и мессенджеров: ВКонтакте обогнал TikTok и WhatsApp по времени, которое россияне проводят в интернете // vk.com URL: <https://vk.com/press/mediascope-october-2022> (дата обращения: 02.05.2024).

15. Как люди образуют связи. Сообщество или социальная сеть 2 // Хабр URL: <https://habr.com/ru/companies/darudar/articles/138014/> (дата обращения: 23.05.2024).
16. Левин Л.М. Социальные сети: основные понятия, характеристики и современные исследования // Проблемы современного образования. – 2019. – № 4. URL: <https://cyberleninka.ru/article/n/sotsialnye-seti-osnovnye-ponyatiya-harakteristiki-i-sovremennye-issledovaniya> (дата обращения: 27.05.2024).
17. Милова Е.А. Влияние социальных сетей на психологию личности // Интерэкспо Гео-Сибирь. – 2012. – № 6. URL: <https://cyberleninka.ru/article/n/vliyanie-sotsialnyh-setey-na-psihologiyu-lichnosti> (дата обращения: 27.05.2024).
18. Ограничение Messages API. VK API Roadmap. // dev.vk.com URL: <https://dev.vk.com/ru/reference/roadmap#%D0%9E%D0%B3%D1%80%D0%B0%D0%BD%D0%B8%D1%87%D0%B5%D0%BD%D0%B8%D0%B5%20Messages%20API> (дата обращения 03.06.2024).
19. Орлова Т.П. К вопросу об определении понятия "цифровая коммуникация" и способах её изучения // Вестник магистратуры. – 2020. – № 6 (105). URL: <https://cyberleninka.ru/article/n/k-voprosu-ob-opredelenii-ponyatiya-tsifrovaya-kommunikatsiya-i-sposobah-ee-izucheniya> (дата обращения: 05.05.2024).
20. Основы проектирования архитектуры простой социальной сети // Хабр URL: <https://habr.com/ru/companies/otus/articles/765014/> (дата обращения: 21.05.2024).
21. От слабых связей к сильным. Сообщество или социальная сеть 3 // Хабр URL: <https://habr.com/ru/companies/darudar/articles/138217/> (дата обращения: 24.05.2024).
22. Плутотаренко С.А. Рунет сегодня. Аналитика. Цифры. Факты. // Годовая расширенная коллегия Минкомсвязи России. - Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации, 2015. URL: <https://digital.gov.ru/uploaded/presentations/kollegiya-mks-sergejplugotarenkopresentation1920x1080ver3.pdf> (дата обращения: 02.06.2024). URL страницы события: <https://digital.gov.ru/ru/events/media/963/> (дата обращения: 02.06.2024).
23. Прончев Г.Б. Трансформация моделей социальной коммуникации в цифровую эпоху // Вестник Московского университета. Серия 18. Социология и политология. – 2022. – № 4. URL: <https://cyberleninka.ru/article/n/transformatsiya-modeley-sotsialnoy-kommunikatsii-v-tsifrovuyu-epohu> (дата обращения: 26.05.2024).
24. Поддержание сильных связей. Сообщество или социальная сеть 4 // Хабр URL: <https://habr.com/ru/companies/darudar/articles/138483/> (дата обращения: 26.05.2024).
25. Почему Россия – страна четвертого мира в социальных сетях // Хабр URL: <https://habr.com/ru/articles/151793/> (дата обращения: 01.06.2024).
26. Рубинштейн А.Я., Соколова Е.К., Дудкина Е.А. Гражданское общество, социальные сети и культурная активность в цифровой среде // Пространство экономики. – 2022. – № 2. URL: <https://cyberleninka.ru/article/n/grazhdanskoe-obschestvo-sotsialnye-seti-i-kulturnaya-aktivnost-v-tsifrovoy-srede> (дата обращения: 06.06.2024).
27. Сердюкова Елена Александровна. Влияние интернета и социальных сетей на жизнь современного человека // Science Time. – 2021. – № 12 (96). URL: <https://cyberleninka.ru/article/n/vliyanie-interneta-i-sotsialnyh-setey-na-zhizn-sovremennogo-cheloveka> (дата обращения: 11.06.2024).

28. Сообщество или социальная сеть? // Хабр URL: <https://habr.com/ru/companies/darudar/articles/137825/> (дата обращения: 29.05.2024).
29. Социальная сеть, как инструмент преобразования человека и общества (light edition) // Хабр URL: <https://habr.com/ru/articles/361851/> (дата обращения: 29.05.2024).
30. Социальная сеть «МирТесен.ру» открылась для публичного тестирования // Хабр URL: <https://habr.com/ru/articles/13781/> (дата обращения: 29.05.2024).
31. Стало известно, сколько времени россияне тратят на интернет // gazeta.ru URL: <https://www.gazeta.ru/tech/news/2023/09/07/21240146.shtml> (дата обращения: 27.04.2024)
32. Толковый словарь русского языка : 80000 слов и фразеологических выражений / С. И. Ожегов, Н. Ю. Шведова; РАН, Ин-т русского языка им. В. В. Виноградова. – 4-е изд., доп. – Москва : Азбуковник, 1997, 1999, 2001, 2003. – 943 с. – ISBN 5-89285-003-X.
33. Численность населения Российской Федерации по муниципальным образованиям на 1 января 2023 : Статистический сборник / Ред. колл.: Алексеева В.С., Гильманов Р.И. : Росстат, 2023. URL: <https://rosstat.gov.ru/compendium/document/13282> (дата обращения: 06.06.2024).
34. API VKontakte “API VKontakte. It is an interface that allows you to get information from the database vk.com HTTP requests to a special server. The syntax of requests and the type of data they return are strictly defined on the side of the service itself.” – 2024. – № 5.199 // VK; VK for developers URL: <https://dev.vk.com/ru/api/overview>. (дата обращения: 02.05.2024).
35. Creative Commons. Documentation. URL: <https://wiki.creativecommons.org/wiki/documentation> (дата обращения: 10.06.2024).
36. DOI.org. What is a DOI? URL: <https://www.doi.org/the-identifier/what-is-a-doi/> (дата обращения: 10.06.2024).
37. Harvard University. Citation Management Tools. URL: <https://usingsources.fas.harvard.edu/citation-management-tools> (дата обращения: 10.06.2024).
38. TDLlib. “TDLlib (Telegram Database library) is a cross-platform library for building Telegram clients. It can be easily used from almost any programming language.” – 27.06.2024. – № 1.8.30 // Telegram Library. URL: <https://core.telegram.org/tldlib> (дата обращения: 03.05.2024).
39. Unified Modeling Language. “A specification defining a graphical language for visualizing, specifying, constructing, and documenting the artifacts of distributed object systems.” – 17.12.2017. – No. 2.5.1 // Object Management Group (OMG). URL: <https://www.omg.org/spec/UML/2.5.1/PDF> (дата обращения: 15.05.2024).
40. VkNet. “Vkontakte API for .NET” – 02.02.2024. – № 1.78.0 // vknet. URL: <https://github.com/vknet/vk>, <https://vknet.github.io/vk/> (дата обращения: 06.05.2024).
41. WTelegramClient. “This library allows you to connect to Telegram and control a user programmatically (or a bot, but WTelegramBot is much easier for that). All the Telegram Client APIs (MTPProto) are supported so you can do everything the user could do with a full Telegram GUI client.” – 28.06.2024. – № 4.1.1 // wiz0u. URL: <https://wiz0u.github.io/WTelegramClient/> (дата обращения: 29.04.2024).
42. .NET documentation // Microsoft URL: <https://learn.microsoft.com/en-gb/dotnet/> (дата обращения: 11.05.2024).
43. .NET supported versions of operating systems of Windows, Linux families. .NET 8 - Supported OS versions. URL: <https://github.com/dotnet/core/blob/main/release-notes/8.0/supported-os.md> (дата обращения: 09.06.2024).

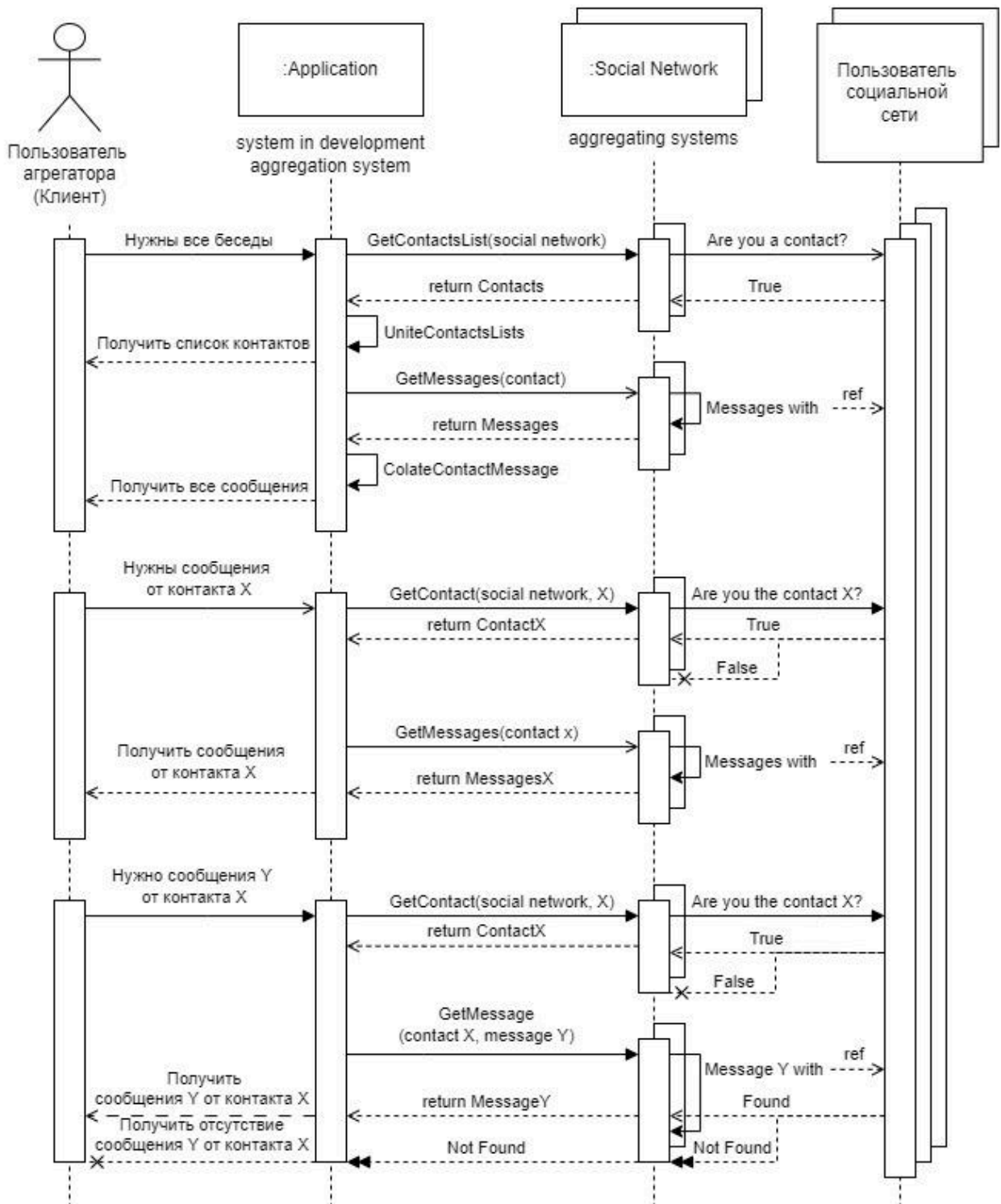




Приложение В

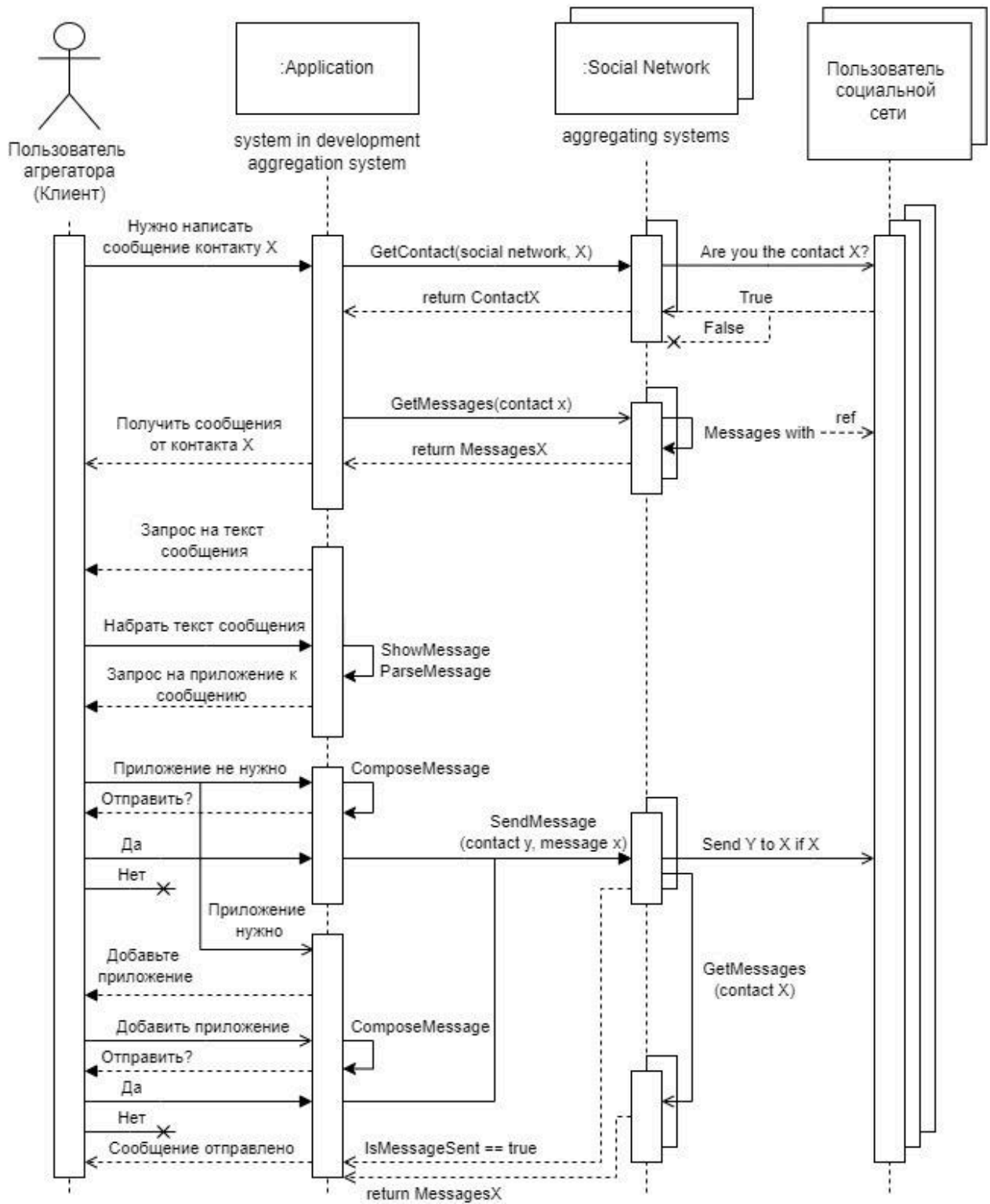
Вариант использования 1.1

Беседа — чат с определенным контактом (контактами).



Приложение Г

Вариант использования 2.1





Приложение Е

Тип для отображения социальной сети и ее API

```
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.Configuration.Json;
using TL;
using WTelegram;
using VkNet;
using VkNet.Model;
using VkNet.Enums.Filters;
using SocialNetworksManager.Configuration;

namespace SocialNetworksManager.Sources
{
    interface ISocialNetwork
    {
        static abstract Guid Id { get; }
        static abstract string? Name { get; }
        static abstract Color BaseColor { get; }
        static abstract Color SecondColor { get; }
        static abstract void Connect();
        static abstract void Disconnect();
        static abstract void GetContacts();
        static abstract Task<List<MessageWrap>> GetMessages(long APIId, int offset, int limit, string contactType);
        static abstract Task<bool> SendMessage(long user, MessageUnwrap msg, string contactType);
    }

    class TelegramSocialNetwork : ISocialNetwork
    {
        public static Guid Id { get; } = new Guid();
        public static string? Name { get; } = "Telegram";
        public static Color BaseColor { get; } = Color.SkyBlue;
        public static Color SecondColor { get; } = Color.White;
        private static Client? apiClient;
        private static IConfigurationRoot? aggregationConfiguration;
        private static TgApiConfig? tgApiConfiguration;

        public static void Connect()
        {
            GetAggregationConfiguration();
            apiClient = new Client(ParseTgConfiguration);
        }

        public static void Disconnect() {}
        static void ISocialNetwork.GetContacts(){}

        private static string? ParseTgConfiguration(string argument)
        {
            return argument switch
            {
                "api_id" => tgApiConfiguration?.ApiId,
                "api_hash" => tgApiConfiguration?.ApiHash,
                "phone_number" => tgApiConfiguration?.Phone,
                "verification_code" => tgApiConfiguration?.Code,
                "first_name" => tgApiConfiguration?.FirstName,
                "last_name" => tgApiConfiguration?.LastName,
                "password" => tgApiConfiguration?.Password,
                "server_address" => tgApiConfiguration?.Server,
                _ => null,
            };
        }

        private static void GetAggregationConfiguration()
        {
            var builder = new ConfigurationBuilder().
                SetBasePath(Directory.GetCurrentDirectory()).
                AddJsonFile("Configs/aggregation.json", optional: false, reloadOnChange: true);
            aggregationConfiguration = builder.Build();
            tgApiConfiguration = aggregationConfiguration.GetSection("TgConfig").Get<TgApiConfig>();
        }

        public static async Task<(IEnumerable<TL.User> users, IEnumerable<ChatBase> chats, Dictionary<long, MessageBase> lastMessages)> GetAllContacts()
```

```
{
    _ = await apiClient.LoginUserIfNeeded();

    var dialogs = await apiClient.Messages_GetAllDialogs();
    var userList = new List<TL.User>();
    var chatsList = new List<TL.Chat>();
    var lastMessagesByldDict = new Dictionary<long, MessageBase>();

    foreach (var dialog in dialogs.dialogs)
    {
        switch (dialogs.UserOrChat(dialog))
        {
            case TL.User user when user.IsActive:
                userList.Add(user);

                var lastMesgForUser = await apiClient.Messages_GetHistory(user, limit: 1);
                if (lastMesgForUser.Messages.Length > 0)
                {
                    lastMessagesByldDict[user.ID] = lastMesgForUser.Messages[0];
                }

                break;

            case TL.Chat chat when chat.IsActive:
                chatsList.Add(chat);

                var lastMesgForChat = await apiClient.Messages_GetHistory(chat, limit: 1);
                if (lastMesgForChat.Messages.Length > 0)
                {
                    lastMessagesByldDict[chat.ID] = lastMesgForChat.Messages[0];
                }

                break;
        }
    }
    return (userList, chatsList, lastMessagesByldDict);
}

public static async Task<List<MessageWrap>> GetMessages(long APIId, int offset, int limit, string contactType)
{
    _ = await apiClient.LoginUserIfNeeded();
    InputPeer peer;
    var totalFetched = 0;

    if (contactType == "userTG")
    {
        var dialogs = await apiClient.Messages_GetAllDialogs();
        peer = dialogs.users[APIId];
    }
    else
    {
        var chats = await apiClient.Messages_GetAllChats();
        peer = chats.chats[APIId];
    }

    return await FetchMessages(offset, limit, peer, totalFetched);
}

private static async Task<List<MessageWrap>> FetchMessages(int offset, int limit, InputPeer peer, int totalFetched)
{
    List<MessageWrap> messagesList = new List<MessageWrap>();

    for (int add_offset = offset; totalFetched < limit;)
    {
        var history = await apiClient.Messages_GetHistory(peer, offset_id: 0, add_offset: add_offset, limit: limit - totalFetched);
        if (history.Messages.Length == 0) break;

        foreach (var msgBase in history.Messages)
        {
            var from = history.UserOrChat(msgBase.From ?? msgBase.Peer);
            char flag = 'u';

            if (from.ID != apiClient.UserId)
            {
                flag = 'i';
            }
            else
            {
                flag = 'u';
            }
            messagesList.Add(new MessageWrap(msgBase, flag));
            totalFetched++;
        }
        add_offset = history.Messages.Last().ID + 1;
    }
    return messagesList;
}
```

```
        {
            flag = 'o';
        }

        if (msgBase is TL.Message msg)
        {
            AnyMessageType mt = new AnyMessageType(msg.message + " " + msg.media, 'p', msgBase.Date.ToLocalTime());
            MessageWrap mw = new(msgBase.ID, flag, mt);
            messagesList.Add(mw);
        }
        else if (msgBase is MessageService ms)
        {
            AnyMessageType mt = new AnyMessageType(ms.action.GetType().Name[13..], 's', ms.date.ToLocalTime());
            MessageWrap mw = new(ms.ID, flag, mt);
            messagesList.Add(mw);
        }
    }

    int fetchedCount = history.Messages.Length;
    totalFetched += fetchedCount;
    add_offset += fetchedCount;
}

return messagesList;
}

public static async Task<bool> SendMessage(long user, MessageUnwrap msg, string contactType)
{
    if (contactType == "userTG")
    {
        var dialogs = await apiClient.Messages_GetAllDialogs();
        await apiClient.SendMessageAsync(dialogs.users[user], msg.Data);
        return true;
    }
    else if (contactType == "chatTG")
    {
        var chats = await apiClient.Messages_GetAllChats();
        await apiClient.SendMessageAsync(chats.chats[user], msg.Data);
        return true;
    }

    return false;
}

}

class VKSocialNetwork : ISocialNetwork
{
    public static Guid Id { get; } = new Guid();
    public static string? Name { get; } = "VK";
    public static Color BaseColor { get; } = Color.MediumBlue;
    public static Color SecondColor { get; } = Color.White;
    private static VkApi? apiClient;
    private static IConfigurationRoot? aggregationConfiguration;
    private static VkApiConfig? vkApiConfiguration;

    public static void Connect()
    {
        GetVkConfiguration();
        apiClient = new VkApi();
        apiClient.Authorize(ParseVkConfiguration());
    }

    public static void Disconnect() {}
    private static ApiAuthParams? ParseVkConfiguration() {}

    public static void GetVkConfiguration()
    {
        var builder = new ConfigurationBuilder().
            SetBasePath(Directory.GetCurrentDirectory()).
            AddJsonFile("Configs/aggregation.json", optional: false, reloadOnChange: true);

        aggregationConfiguration = builder.Build();
        vkApiConfiguration = aggregationConfiguration.GetSection("VkConfig").Get<VkApiConfig>();
    }

    public static async Task<(IEnumerable<VkNet.Model.User> friends, Dictionary<long, DateTime?> lastMessages)>
    GetContacts()
```

```

    {
        var lastMessagesByldDict = new Dictionary<long, DateTime?>();
        Random randomGen = new Random();
        DateTime startRandom = new DateTime(2023, 8, 1);
        var friends = await apiClient.Friends.GetAsync(new FriendsGetParams
        {
            UserId = vkApiConfiguration.UserId,
            Fields = ProfileFields.All
        });

        foreach (var friend in friends)
        {
            var lastMessageTimes = GetLastMessageTime(randomGen, startRandom);
            lastMessagesByldDict[friend.Id] = lastMessageTimes;
        }

        return (friends, lastMessagesByldDict);
    }

    private static DateTime? GetLastMessageTime(Random gen, DateTime start){}
    static Task<List<MessageWrap>> ISocialNetwork.GetMessages(long APIId, int offset, int limit, string contactType){}
    static Task<bool> ISocialNetwork.SendMessage(long user, MessageUnwrap msg, string contactType){}
    static void ISocialNetwork.GetContacts(){}

}

class OkSocialNetwork : ISocialNetwork{}
class ViSocialNetwork : ISocialNetwork{}

```

Тип для отображения контактов внутри системы при получении данных из социальной сети

```

namespace SocialNetworksManager.Sources
{
    interface IContact
    {
        Guid Id { get; set; }
        string? ContactName { get; set; }
        ISocialNetwork? SocialNetwork { get; }
        List<MessageWrap>? MessagesList { get; set; }
        DateTime? LastMessageTime { get; set; }
        string? AgoMessageTime { get; set; }
        DateTime? LastOnlineTime { get; set; }
        string? SideColor { get; }
        long APIId { get; set; }
        string? ContactType { get; set; }
        public byte[] ProfilePicture { get; set; }

        Task<List<MessageWrap>> GetInitialPackMessages();
        List<MessageWrap> GetAllPackMessage();
        MessageWrap GetLastMessage();
        MessageWrap GetMessageAt(int index);
        MessageUnwrap CreateMessage(IMessage msg);
        Task<bool> SendMessage(MessageUnwrap msgWrap, long userId);
    }

    class AnyContactType : IContact
    {
        public Guid Id { get; set; } = new Guid();
        public string? ContactName { get; set; }
        public ISocialNetwork? SocialNetwork { get; }
        public List<MessageWrap>? MessagesList { get; set; }
        public DateTime? LastMessageTime { get; set; }
        public string? AgoMessageTime { get; set; }
        public DateTime? LastOnlineTime { get; set; }
        public string? SideColor { get; set; }
        public long APIId { get; set; }
        public string? ContactType { get; set; }
        public byte[]? ProfilePicture { get; set; }

        public AnyContactType(int DefaultType = 1){}

        public MessageUnwrap CreateMessage(IMessage msg)
        {
            return new MessageUnwrap(-1, 'o', (AnyMessageType)msg);
        }
    }
}

```

```
public async Task<bool> SendMessage(MessageUnwrap mesg, long userId)
{
    if (ContactType == "chatTG")
    {
        await TelegramSocialNetwork.SendMessage(userId, mesg, ContactType);
        return true;
    }
    else if (ContactType == "userTG")
    {
        await TelegramSocialNetwork.SendMessage(userId, mesg, ContactType);
        return true;
    }
    return false;
}

public List<MessageWrap> GetAllPackMessage() {}
public MessageWrap GetLastMessage() {}
public MessageWrap GetMessageAt(int index) {}

public async Task<List<MessageWrap>> GetInitialPackMessages()
{
    return await TelegramSocialNetwork.GetMessages(APIId, 0, 30, ContactType);
}
}
```

Тип для отображения сообщений при их получении из социальной сети через API

```
namespace SocialNetworksManager.Sources
{
    interface IMessage
    {
        Guid Id { get; }
        string? Data { get; }
        char? Flag { get; }
        DateTime? MessageTime { get; }
        List<string>? Attachments { get; }
        void AddAttachment(string Attachment);
        void RemoveAttachment(string Attachment);
    }

    class AnyMessageType : IMessage
    {
        public Guid Id { get; } = new Guid();
        public string? Data { get; }
        public char? Flag { get; }
        public DateTime? MessageTime { get; }
        public List<string>? Attachments { get; }

        public AnyMessageType(string _Data, char _Flag, DateTime _Time) {
            Data = _Data;
            Flag = _Flag;
            MessageTime = _Time;
        }

        void IMessage.AddAttachment(string Attachment){}
        void IMessage.RemoveAttachment(string Attachment){}
    }

    class MessageWrap
    {
        public int Id { get; }
        public char OuterFlag { get; set; }
        public AnyMessageType Message { get; set; }

        public MessageWrap(int _Id, char _OuterFlag, AnyMessageType _Message)
        {
            Id = _Id;
            OuterFlag = _OuterFlag;
            Message = _Message;
        }
    }
}
```

Приложение Ж

Функция для обновления списка контактов при их получении из API

```
private async Task UpdateContacts()
{
    var (users, chats, lastMessages) = await TelegramSocialNetwork.GetAllContacts();
    var (friends, lastMessagesTimeVK) = await VKSocialNetwork.GetContacts();

    var newContacts = new List<AnyContactType>();

    newContacts.AddRange(users.Select(user => new AnyContactType
    {
        ContactName = user.first_name + " " + user.last_name,
        SideColor = TelegramSocialNetwork.BaseColor.ToString(),
        AgoMessageTime = TimeAgo(lastMessages[key: user.ID].Date),
        LastMessageTime = lastMessages[key: user.ID].Date,
        APIId = user.ID,
        ContactType = "userTG",
    }));

    newContacts.AddRange(chats.Select(chat => new AnyContactType
    {
        ContactName = chat.Title,
        SideColor = TelegramSocialNetwork.BaseColor.ToString(),
        AgoMessageTime = TimeAgo(lastMessages[key: chat.ID].Date),
        LastMessageTime = lastMessages[key: chat.ID].Date,
        APIId = chat.ID,
        ContactType = "chatTG",
    }));

    newContacts.AddRange(friends.Select(friend => new AnyContactType
    {
        ContactName = friend.FirstName + " " + friend.LastName,
        SideColor = VKSocialNetwork.BaseColor.ToString(),
        AgoMessageTime = TimeAgo(lastMessagesTimeVK[key: friend.Id]),
        LastMessageTime = lastMessagesTimeVK[key: friend.Id],
        APIId = friend.Id,
        ContactType = "friendVK",
    }));

    newContacts = newContacts.OrderByDescending(c => c.LastMessageTime).ToList();

    ActiveContacts.Clear();
    ActiveContacts.AddRange(newContacts);
    ActiveContacts.Add(new AnyContactType(4));
    anyContactTypeBindingSource.DataSource = null;
    anyContactTypeBindingSource.DataSource = ActiveContacts;
    anyContactTypeBindingSource.ResetBindings(false);
    ContactsGridView.Refresh();
}
```

Функция для получения списка сообщений при их получении из API

```
private async Task LoadMessages(object sender, EventArgs e)
{
    var newMessages = new List<MessageUnwrap>();
    AnyContactType? selectedContact = ActiveContacts.FirstOrDefault(c => c.Id.ToString() == selectedId);
    List<MessageWrap> ActiveMessageWrapped = await selectedContact.GetInitialPackMessages();

    ActiveMessages.Clear();

    newMessages.AddRange(ActiveMessageWrapped
        .Select(msgWrap => new MessageUnwrap(
            msgWrap.Id,
            msgWrap.OuterFlag,
            msgWrap.Message
        )).ToList());

    ActiveMessages = newMessages.OrderByDescending(c => c.Time).Reverse().ToList();
    ActiveMessages.Add(new MessageUnwrap());
    ActiveMessages = ActiveMessages
        .Where(msg => !string.IsNullOrEmpty(msg.Data))
```

<pre> .ToList(); messageUnwrapBindingSource.DataSource = null; messageUnwrapBindingSource.DataSource = ActiveMessages; messageUnwrapBindingSource.ResetBindings(false); LabelLastMessageButton_Click(sender, e); } </pre>
<p>Функция для создания и отправки сообщения через API</p>
<pre> private void LabelSendMessageButton_Click(object sender, EventArgs e) { var msg = CreateMessage(); var selectContact = ActiveContacts.FirstOrDefault(item => item.Id.ToString() == ContactsGridView.SelectedRows[0].Cells[4].Value.ToString()); var msgUnwrap = selectContact.CreateMessage(msg); var userId = ContactsGridView.SelectedRows[0].Cells[5].Value.ToString(); selectContact.SendMessage(msgUnwrap, long.Parse(userId)); LoadMessages(sender, e); } </pre>
<p>Функция для поиска контактов среди списка контактов</p>
<pre> private void TextBoxSearchContact_TextChanged(object sender, EventArgs e) { if (textBoxSearchContact.Text == "Поиск ☐" textBoxSearchContact.Text == "☐" ContactsGridView.RowCount <= 0) { return; } var searchText = textBoxSearchContact.Text.ToLower(); var filteredList = ActiveContacts.Where(item => item.ContactName.ToLower().Contains(searchText)).ToList(); filteredList.Add(new AnyContactType(4)); anyContactTypeBindingSource.DataSource = filteredList; ContactsGridView.Refresh(); if (ContactsGridView.RowCount > 0) { ContactsGridView.Rows[ContactsGridView.Rows.Count - 1].Selected = true; ContactsGridView.Rows[ContactsGridView.Rows.Count - 1].Visible = false; } } </pre>
<p>Класс main приложения, с полями, конструктором и инициализаторами</p>
<pre> public partial class MainForm : Form { private readonly Color _foreColorPrimary = Color.FromArgb(247, 148, 29); private readonly Color _foreColorSecond = Color.FromArgb(0, 0, 0); private readonly Color _foreColorThird = Color.FromArgb(139, 0, 0); private readonly Color _backColorPrimary = Color.FromArgb(242, 242, 242); private readonly Color _backColorSecond = Color.FromArgb(255, 255, 255); private List<AnyContactType> ActiveContacts = []; private List<MessageUnwrap> ActiveMessages = []; private static IConfigurationRoot? statsConfiguration; private static StatsConfig? statsMapConfiguration; private static DataGridViewRow? selectedRow; private static string? selectedId; private static System.Timers.Timer _timer; private static readonly Random _random = new Random(); public MainForm() { InitializeComponent(); GetStatsConfiguration(); InitialParseConfigs(); ParseStatsToLabel(); } private void MainForm_Load(object sender, EventArgs e) </pre>

```
{
    TelegramSocialNetwork.Connect();
    VKSocialNetwork.Connect();
    //OkSocialNetwork.Connect();
    //ViSocialNetwork.Connect();

    InitialLoadContacts();
}

private async void InitialLoadContacts()
{
    await UpdateContacts();
    if (ContactsGridView.RowCount > 0)
    {
        ContactsGridView.Rows[ContactsGridView.Rows.Count - 1].Selected = true;
        ContactsGridView.Rows[ContactsGridView.Rows.Count - 1].Visible = false;
    }
}
}
```