Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Российский государственный профессионально-педагогический университет»

TELEGRAM-БОТ ДЛЯ ПРОДВИЖЕНИЯ БИЗНЕС-АККАУНТОВ В СОЦИАЛЬНОЙ СЕТИ INSTAGRAM

Выпускная квалификационная работа
по направлению подготовки 09.03.02 «Информационные системы и технологии»
профилю подготовки «Информационные технологии в медиаиндустрии»

Идентификационный номер ВКР: 151

Екатеринбург 2018

Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Российский государственный профессионально-педагогический университет»
Институт инженерно-педагогического образования
Кафедра информационных систем и технологий

К ЗАЩИТЕ ДОПУСКАЮ
Заведующая кафедрой ИС
_____ Н. С. Толстова
«____» _____ 2018 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

TELEGRAM-БОТ ДЛЯ ПРОДВИЖЕНИЯ БИЗНЕС-АККАУНТОВ В СОЦИАЛЬНОЙ СЕТИ INSTAGRAM

Исполнитель:
обучающийся группы ИТм-402 Д. Н. Шулепов
Руководитель:
канд. пед. наук, доцент Н. С. Толстова
Нормоконтролер: Н. В. Хохлова

Екатеринбург 2018

АННОТАЦИЯ

Выпускная квалификационная работа состоит из Telegram-бота и пояснительной записки на 60 страницах, содержащей 68 рисунков, 1 таблицу, 30 источников литературы, а также 1 приложение на 2 листах.

Ключевые слова: TELEGRAM-BOT, INSTAGRAM, БИЗНЕС-АККАУНТ, СОЦИАЛЬНАЯ СЕТЬ.

Шулепов Д. Н. Telegram-бот для продвижения бизнес-аккаунтов в социальной сети Instagram: выпускная квалификационная работа / Д. Н. Шулепов ; Рос. гос. проф.-пед. ун-т, Ин-т инж.-пед. образования, Каф. информ. систем и технологий. — Екатеринбург, 2018. — 60 с.

В работе рассмотрены вопросы по разработке Telegram-бота для автоматизации действий, направленных на привлечение внимания пользователей социальной сети к своему профилю.

Целью работы являлось разработка Telegram-бота, осуществляющего автоматизацию определенных действий, направленных на привлечение внимания к профилю в социальной сети. Для достижения поставленной цели были решены такие основные задачи, как: проведен анализ предметной области; проанализированы существующие решения, обеспечивающие подобный функционал; проведен анализ средств для разработки Telegram-ботов.

Практическая значимость данной работы заключалась в использовании результатов работы в практической деятельности по привлечению клиентов к проектам.

Разработка Telegram-бота прошла без технологических сбоев. Тестирование проводилось на локальном компьютере, после положительных результатов и оценок проект будет перенесен в глобальную среду.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Сейчас социальные сети уже давно не просто служат для организации взаимодействий пользователей, но и являются мощным рекламным инструментом, которым пользуются все, начиная от небольших частных компаний и заканчивая такими организациями как National Geographic, Nasa, Lamborghini, Nike, Gucci, Louis Vuitton и многими другими.

Согласно сайту www.marketing.spb.ru, «наружную рекламу уже опережает интернет-реклама. Инструменты интернет-рекламы позволяют точечно влиять на нужные заказчику аудитории, выстраивать со своим клиентом более близкие и эффективные отношения. Размещая рекламу в Интернете, рекламодатель имеет возможность получать ясные, актуальные отчеты о количестве людей, которые совершили запланированные рекламной кампанией действия, возможность быстрого реагирования на полученные результаты также увеличивают эффективность интернет-рекламы. Добавим в этот список интерактивную коммуникацию, достоинства геотаргетирования и самую низкую цену за один контакт на всем рынке рекламы» [14]. Таким образом интернет-реклама является обязательным инструментом как для только что созданных проектов, так и для проектов, которые только начинают свое продвижение в Интернете. И поэтому вопросы, связанные с применением интернет-рекламы как никогда актуальны.

Объект выпускной квалификационной работы — процесс привлечения внимания пользователей социальной сети к проекту.

Предмет выпускной квалификационной работы — возможности привлечения внимания пользователей социальной сети к проекту с помощью программного продукта автоматизации взаимодействия пользователей.

Цель выпускной квалификационной работы — разработать Telegram-бот, для автоматизации действий, направленных на привлечение внимания пользователей социальной сети.

Задачи выпускной квалификационной работы:

1. Проанализировать литературные и интернет-источники с целью выявления технологий, способов и средств привлечения внимания пользователей социальных сетей.

2. Выявить возможности применения программ-ботов для привлечения внимания пользователей социальных сетей.

3. Сформулировать требования, спроектировать и разработать Telegram-бот для автоматизации действий, направленных на привлечение внимания пользователей социальной сети.

4. Провести апробацию Telegram-бот для привлечения внимания пользователей социальной сети к своему профилю.

1 ПРОГРАММНЫЕ СРЕДСТВА ДЛЯ ПРИВЛЕЧЕНИЯ ВНИМАНИЯ ПОЛЬЗОВАТЕЛЕЙ В СОЦИАЛЬНЫХ СЕТЯХ

1.1 Технология привлечения внимания в социальных сетях

Технология привлечения внимания в социальных сетях — это реклама и различные ее подвиды, такие как конкурсы, розыгрыши и подобное.

Некоторые разновидности интернет-рекламы согласно сайту leader-web.ru [16]:

- Интернет — контекстная реклама;
- Интернет — социальные сети;
- Интернет — контекстная реклама.

Интернет — контекстная реклама.

Достоинства контекстной рекламы:

- быстрый запуск;
- полный контроль расходов;
- статистика эффективности;
- настройка таргетирования — максимум целевых посетителей;
- отсутствие переплаты.

Недостатки контекстной рекламы:

- однообразность;
- ограничения по внешнему отображению;
- необходимость в навыках для самостоятельной реализации.

Интернет — социальные сети.

Достоинства социальных сетей:

- внедрение необычных методов раскрутки — опросы, горячие обсуждения, стимуляция комментирования, конкурсы и т.п.;

- возможность «быть ближе к народу» и открыто представить свою компанию «в лицах»;
- лояльность аудитории;
- реализация «вирусного маркетинга»;
- комплексное воздействие на имидж и узнаваемость;
- возможность исследования демографических данных потенциальных клиентов.

Недостатки социальных сетей:

- уровень конкуренции;
- временные ресурсы;
- необходимость постоянной модернизации и контроля площадки;
- «скромные» инструменты автоматизированной аналитики.

Интернет — поисковое продвижение.

Достоинства поискового продвижения:

- самая выгодная цена;
- перспективы;
- естественное восприятие;
- дополнение к другим видам продвижения в сети;
- контроль эффективности и расчет прибыли;
- продвинутые методы аналитики;
- возможность предсказания сезонности и правильных корректировок;
- сохранение эффекта некоторое время даже после окончания бюджета.

Недостатки поискового продвижения:

- небыстрый старт;
- обязательно наличие сайта;
- необходимость в опыте и специфических знаниях (и это может быть «плюсом» — трудности для конкурентов);

- влияние поисковых систем и уровня конкуренции.

1.2 Способы привлечения внимания в социальных сетях

Основным способом продвижения своей продукции в социальных сетях конечно является покупка рекламы. Помимо приобретения рекламных услуг в самой сети, также можно заказывать ее у популярных в этой сети пользователей, но зачастую цены на это превышают десятки или даже сотни тысяч рублей.

Цены на показ рекламы на сайте youtube.com согласно reklamnoeburo.ru (рисунок 1):

Цены на формат TrueView In-Stream.

Цена	Просмотры	Показы
8 000 руб.	3,5 тыс.–28 тыс.	76 216
20 000 руб.	14 тыс.–56 тыс.	191 044
50 000 руб.	28 тыс.–126 тыс.	478 856
70 000 руб.	42 тыс.–168 тыс.	668 668
100 000 руб.	70 тыс.–238 тыс.	955 248
200 000 руб.	140 тыс.–462 тыс.	1 910 188
500 000 руб.	378 тыс.–1,16 млн	4 775 470
1 000 000 руб.	756 тыс.–2,38 млн	9 552 452

Рисунок 1 — Цены на рекламу на youtube.com

Таргетированная реклама в Интернете (на сайтах, в соцсетях и т. д.) — это такая, где вы можете выбирать, какой аудитории будете её показывать. Пол, возраст, место проживания, интересы — все эти параметры можно настраивать в таргетированной рекламе.

Основные плюсы таргетированной рекламы:

- можно быстро настроить и запустить (не нужно тратить время на переговоры с блоггером);
- нет «человеческого фактора» (блоггер может плохо сделать свою работу или обмануть и не опубликовать рекламный пост совсем);
- гибкая настройка аудитории, которая будет видеть рекламный пост;
- полный контроль сроков показа и того, сколько будет стоить реклама;
- подробная статистика (сколько человек увидели рекламу, сколько перешли по ссылке и т. д.).

Вторым способом является привлечение внимания пользователей вручную. Плюс такого подхода в том, что он бесплатный, но при этом занимает очень много времени. Человеку, заинтересованному в потребителях, приходится самому исследовать и находить потенциальных клиентов в социальной сети, после чего начинать различными способами, такими как оценка контента, комментирование записей и другими, привлекать его на свой профиль.

Есть так же третий способ. Спам — рассылка однотипных рекламных сообщений большим группам пользователей. Но такой способ в наше время вызывает большое количество негативных реакций и уже почти не используются. Такой вид рекламы хоть и увеличит посещаемость, но может также стать причиной множества отрицательных отзывов, что в итоге только ухудшит ситуацию.

Продвижение «под ключ». Этот способ полностью освобождает вас от необходимости самому разбираться в рекламных алгоритмах. Вы платите другим людям или компаниям, которые делают за вас всю работу.

Ниже представлен список услуг одного из таких агентств с сайта fireseo.ru (рисунок 2) [19].

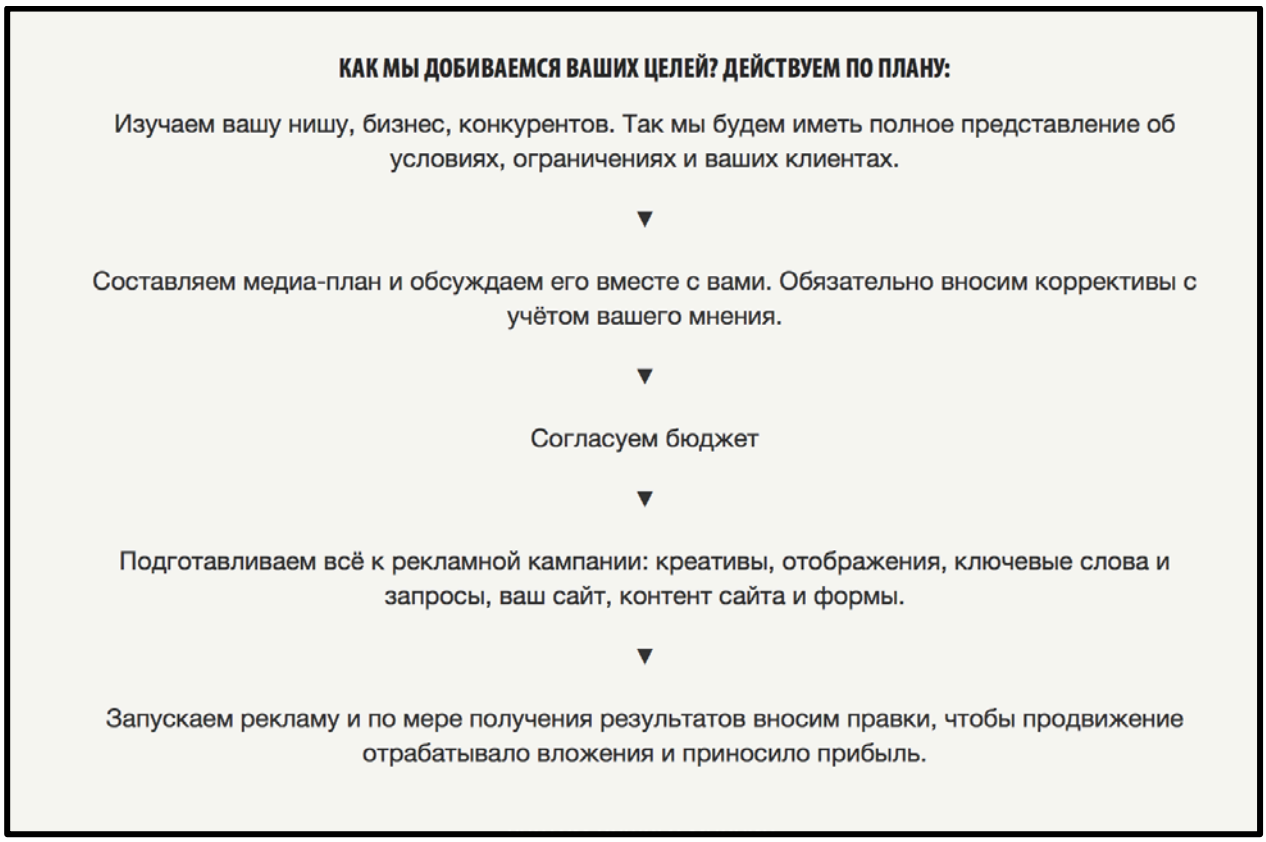


Рисунок 2 — Услуги сайта fireseo.ru

Естественно результат любого из этих способов почти полностью зависит от того, насколько ваш продукт привлекателен для определенной вами аудитории.

1.3 Обзор существующих решений по привлечению внимания пользователей в социальных сетях

На рынке программного обеспечения (ПО) есть множество различных приложений, предоставляющих подобный функционал. Основная проблема таких сервисов — это цена. Вам в любом случае предложат ознакомиться, а затем нужно будет платить либо за каждого привлеченного пользователя, либо за подписку на сервис раз в месяц. Так же обычно такие предложения это или отдельные приложения на смартфон, или веб сайт. К примеру такие сервисы как SocialHammer (рисунок 3), Instaplus (рисунок 5), JetInsta (рисунок 7) обладают всеми вышеперечисленными качествами.

Рисунок 3 — Главная страница SocialHammer

Услуги, предоставляемые SocialHammer [26]:

- пробный период сервиса;
- техподдержка;
- подписка по конкурентам/хештегам/геолокации/списку;
- комментарии по хештегам с возможностью лайкать публикации;
- публикация фотографий с возможностью комментирования;
- отслеживание комментариев и сообщений в директ;
- блокирование неактивных подписчиков.

Цены на услуги SocialHammer представлены ниже (рисунок 4).

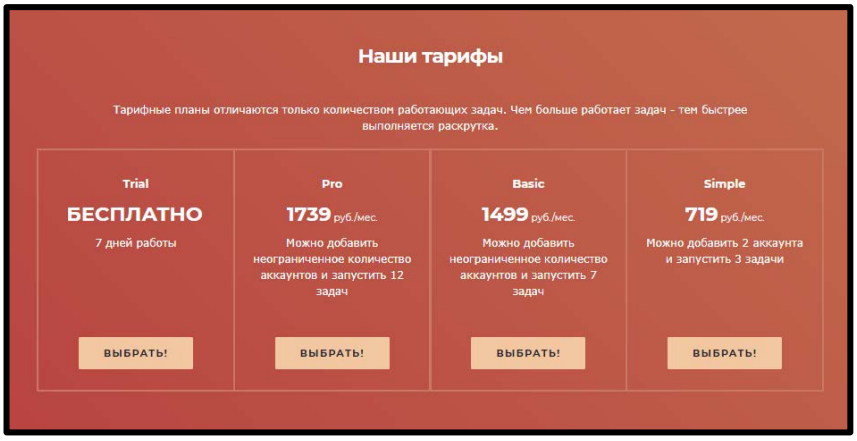


Рисунок 4 — Тарифы SocialHammer

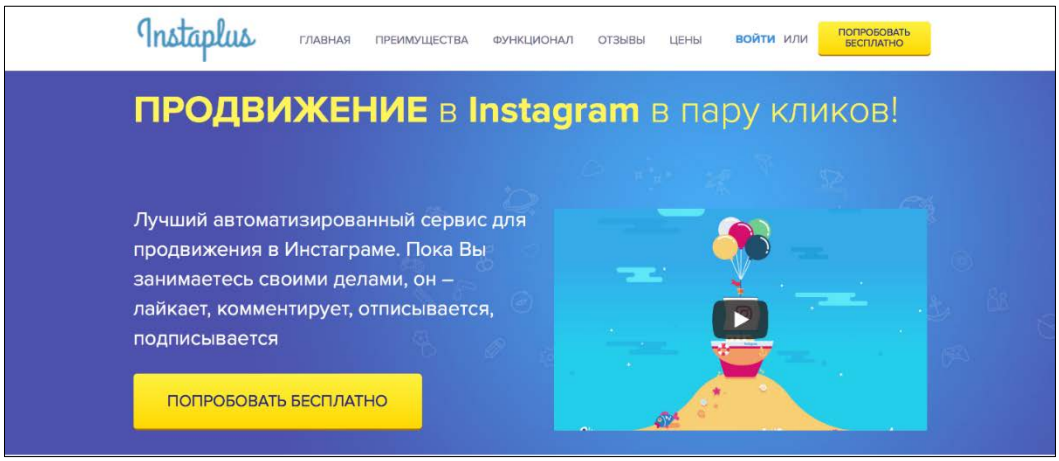


Рисунок 5 — Главная страница Instaplus

Услуги, предоставляемые Instaplus [22]:

- автоматический поиск и проставление «лайков» под фотографиями: по геометкам, по хэштегам, по списку подписчиков ваших конкурентов;
- настройка автоматической отписки от невзаимных подписчиков при достижении лимита;
- настройка автоматического поиска и подписки на аккаунты целевой аудитории по хэштегам и геолокациям, в том числе и на аккаунты подписчиков конкурентов;
- имитация живого общения благодаря созданию и рассылке уникальных комментариев по аккаунтам целевой аудитории;
- бесплатный пробный период.

Цены на услуги Instaplus (рисунок 6).

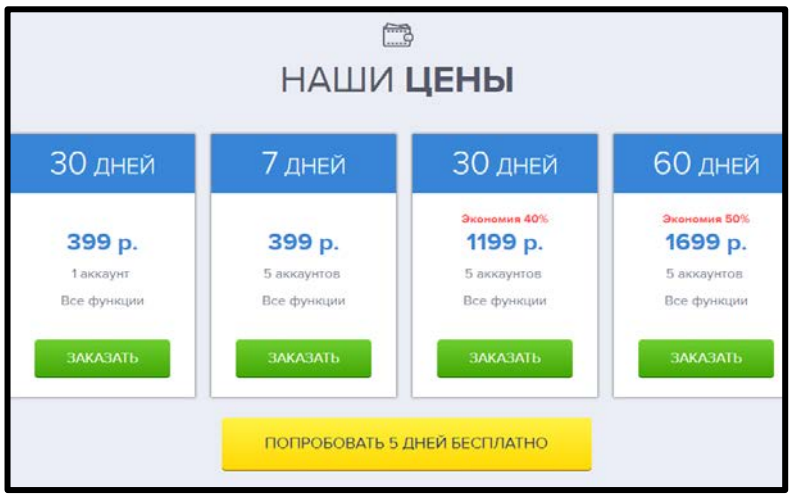


Рисунок 6 — Тарифы Instaplus

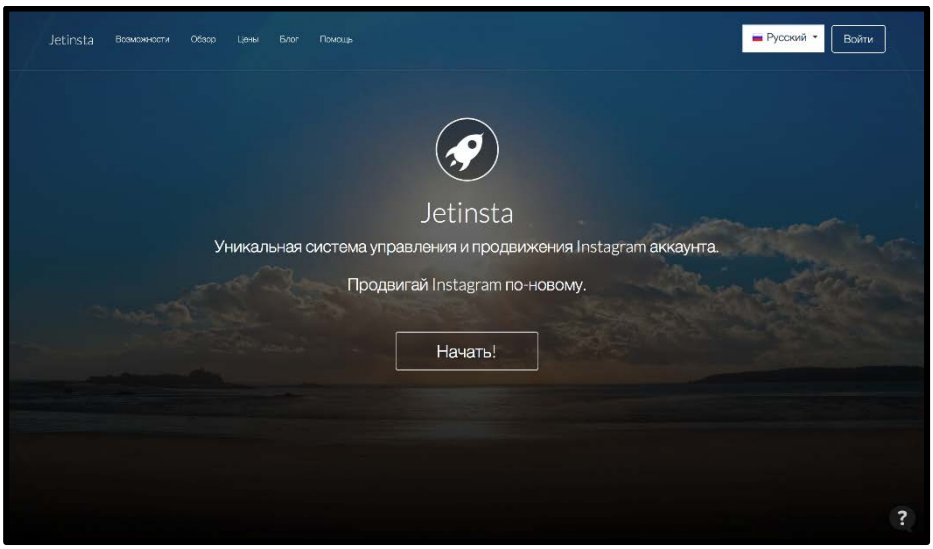


Рисунок 7 — Главная страница JetInsta

Услуги, предоставляемые JetInsta [24]:

- ставьте лайки в интересующих вас аккаунтах автоматически;
- управление несколькими аккаунтами через единый интерфейс;
- слежка за новыми комментариями;
- возможность указывать в фильтре запроса нужные параметры, система отберет наиболее подходящих;
- анализ увеличения числа ваших подписчиков и лайков;
- подбирайте наиболее подходящие хештеги, находите новые, смотрите статистику по ним.

Цены на услуги в сервисе JetInsta (рисунок 8).

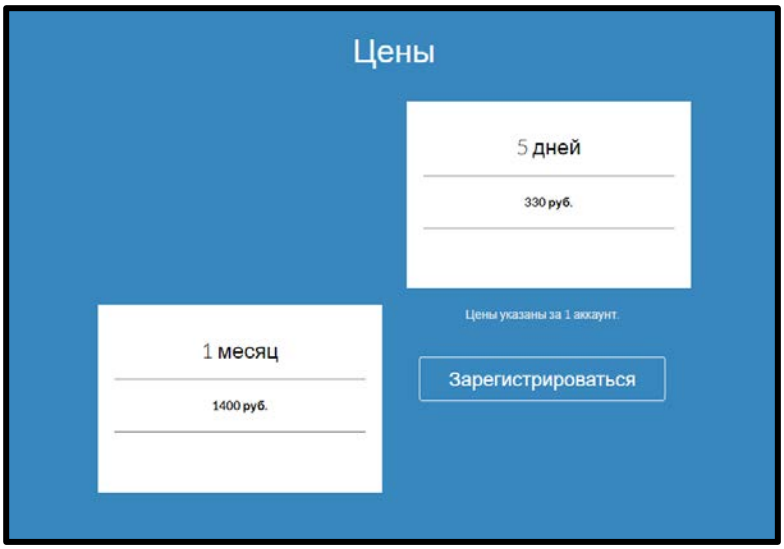


Рисунок 8 — Тарифы JetInsta

Пример компании, предоставляющей продвижение «под ключ» это fireseo.ru (рисунок 9) [19]. Данная компания берет все обязанности по продвижению вашего проекта на себя, лишь учитывая определенные пожелания заказчика.

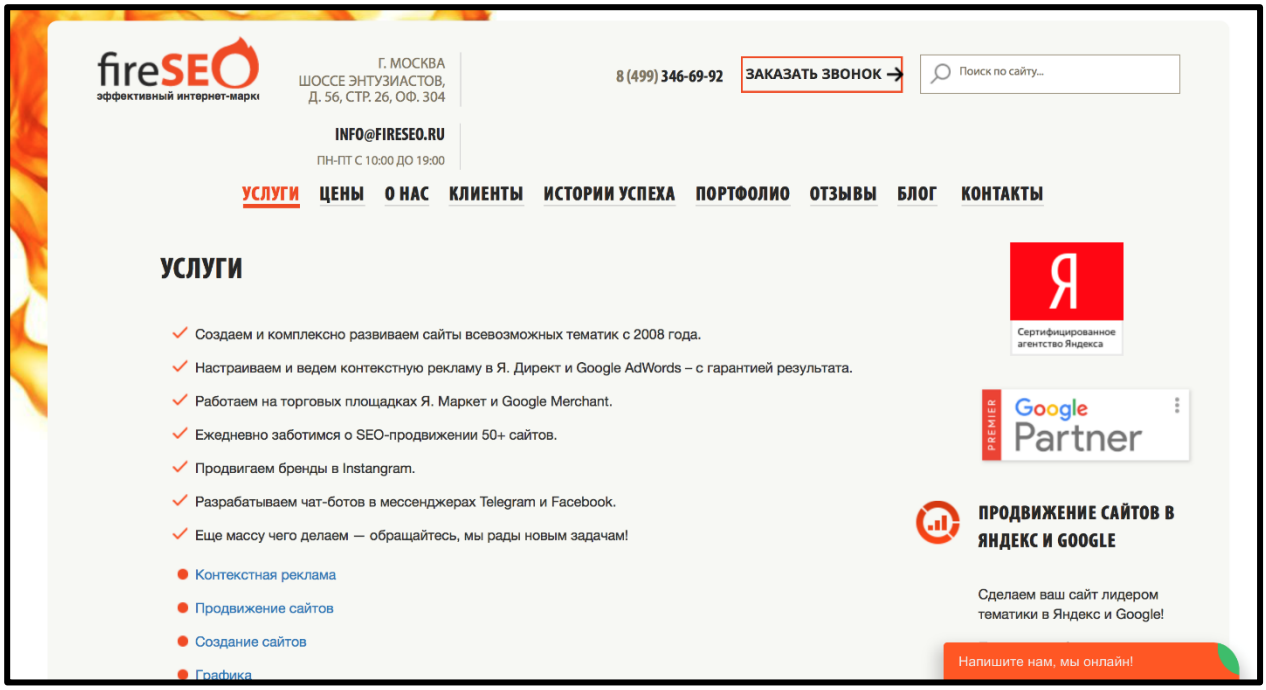


Рисунок 9 — Главная страница fireseo.ru

Среди основных социальных сетей были выделены следующие:

- Facebook;
- ВКонтакте;
- Instagram;
- TwitTer.

Самым простым в вопросах использования, сбора статистики и привлечения пользователей является социальная сеть Instagram.

Instagram — бесплатное приложение для обмена фотографиями и видеозаписями с элементами социальной сети, позволяющее снимать фотографии и видео, применять к ним фильтры, а также распространять их через свой сервис и ряд других социальных сетей [21].

Маркетинговая статистика Instagram согласно mopassan.com:

- 48,8 % брендов присутствуют в Instagram. По прогнозам, эта цифра увеличится до 70,7 %;

- из ста лучших брендов в мире у девяноста есть учётная запись Instagram;
- 96 % фэшн-брендов Соединенных Штатов Америки (США) представлены в Instagram;
- за пределами Китая почти 50 % пользователей Instagram проводят исследования продуктов в социальной сети;
- взаимодействие с брендами в Instagram в 10 раз выше, чем на Facebook, в 54 раза выше, чем в PinTerest, и в 84 раза выше, чем в Twitter;
- более трети пользователей Instagram используют свой мобильный телефон для онлайн-покупок. Таким образом, они на 70 % более склонны к покупкам в сети, чем потребители, не использующие Instagram.

1.4 Обзор платформ для написания программ-ботов

Для реализации проекта решено было остановить свой выбор на среде разработки Microsoft Visual Studio Community 2017, так как она довольно широко распространена, имеет бесплатную версию для использования в образовательных целях и к тому же поддерживает несколько языков.

Модель данной среды использует такие понятия, как решение (solution), проект, пространство имен (namespace) и сборка (assembly). Понятие проекта присутствует во многих средах, например, в среде Delphi. Файл проекта содержит перечисление исходных файлов и прочих ресурсов, из которых система будет строить приложение. В решение среды Visual Studio входят несколько проектов, которые могут быть зависимыми или независимыми друг от друга.

Интегрированная среда разработки обладает большим числом инструментов и функций: может создавать как приложения в консоли, так и программы с графическим интерфейсом, и даже с помощью технологии Windows Forms. Хотя и работает с полным функционалом сразу после уста-

новки, но при отказе одного из компонентов перестаёт работать весь продукт. Удобный интерфейс и лёгкость в понимании работы продукта [17].

В качестве языка программирования выбор пал на C (Си), так как данный язык является объектно-ориентированным, имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML. Относится к семейству языков C-подобным синтаксисом, но исключает некоторые модели, зарекомендовавшие себя как проблематичные при разработке, например, в отличие от C++ не поддерживает множественное наследование классов (между тем допускается множественное наследование интерфейсов).

Для реализации клиентской части необходимо создание и настройка бота в Telegram. Эта возможность реализована внутри самого мессенджера.

Так же для организации работы приложения с Telegram и Instagram необходима API, позволяющие производить различные действия с этими сервисами. Для работы с Telegram был выбран Telegram.Bot API, так как он есть для языка программирования C, и имеет множество обучающих материалов. Для организации взаимодействия с Instagram было принято решение использовать Instasharper, он обладает необходимым набором функций и разработан специально для работы на C.

1.5 Общий алгоритм реализации проекта

Перед написанием кода необходимо изучить всю имеющуюся документацию по теме, обучающие материалы и полезные статьи. После обучения следует написание основы бота. Далее идет подключение основополагающих функций бота, таких как авторизация в Instagram и реакция на сообщения в Telegram. При корректной работе основы начинается период тестиро-

вания имеющихся функций и добавления новых. Когда бот будет полностью работоспособен и протестирован, начинается этап апробирования бота. Описанный алгоритм представлен в Таблица 1.

Таблица 1 — Общий алгоритм реализации проекта

Этап	Действия
1	Изучение материала. Разработка небольших тестовых приложений для понимания логики работы API.
2	Написание консольного приложения с возможностью регистрации в социальной сети Instagram.
3	Наращивание функционала приложения и подготовка к подключению Telegram.Bot API.
4	Создание и настройка бота внутри Telegram для получения уникального ключа доступа и начало создания логики приложения для адекватной работы в рамках Telegram.Bot API.
5	Тестирование и наращивание функционала до удовлетворительного результата.
6	Доработка мелких недочетов и тестирование работоспособности.
7	Апробирование Telegram-бота.

1.6 Обзор возможностей мессенджера Telegram

Telegram — уникальный по своей структуре мессенджер, являющийся кроссплатформенным приложением. Помимо стандартного обмена сообщениями в диалогах и группах, в мессенджере можно хранить неограниченное количество файлов, вести каналы (микроблоги), создавать и использовать ботов. При помощи специального API сторонние разработчики могут создавать боты, специальные аккаунты, управляемые программами. Типичные боты отвечают на специальные команды в персональных и групповых чатах, также они могут осуществлять поиск в интернете или выполнять иные задачи, применяются в развлекательных целях или в бизнесе [29].

Также реализована функция использования socks5 proxy, которая позволяет использовать мессенджер даже в тяжелых для подключения условиях.

SOCKS 4/5 — это распространённое обозначение двух версий сетевого протокола SOCKS 4 и SOCKS 5. Основным преимуществом SOCKS (от. англ.

«sockets» — гнёзда) является возможность работы клиент-серверных приложений за границами межсетевого экранирования, то есть SOCKS прокси может принять запрос от клиента, находящегося за фаерволом, просмотреть его права доступа и передать запрос на внешний сервер.

Telegram-боты — разновидность чат-ботов. Их суть заключается в ответной реакции на определенные сообщения от пользователей. Таким образом, сфера их применения безгранична. Компания Uber, некоторые банки, крупные магазины и многие другие организации используют Telegram-ботов для упрощения и автоматизации внутренних рабочих процессов. При регистрации бота выдается уникальный ключ, с помощью которого в дальнейшем и будет происходить связь между клиентом и сервером. Такая схема исключает необходимость дополнительных настроек клиент-серверной архитектуры, так как все происходит автоматически и занимает несколько строчек кода.

По вышеуказанным причинам Telegram был выбран как самая удобная оболочка для разрабатываемого приложения.

2 СОЗДАНИЕ TELEGRAM-БОТА

2.1 Характеристика заказчика

Заинтересованные в проекте являются недавно открывшиеся компании, или компании, желающие начать привлекать новых клиентов через Instagram. При создании нового проекта, для раскрутки всегда требуются большие вложения в рекламу, которые в наше время все меньше окупаются.

При этом и интернет-реклама может стоить немалых денег. Поэтому тем, кто только начинает свой проект или собирается осваивать интернет-аудиторию и не хочет тратить на рекламу большие средства идеально подойдет решение, позволяющее привлекать к себе внимание пользователей без каких-либо вложений.

В использовании Telegram-бота был заинтересован питомник по продаже котов породы мейн кун Dias MonTes, потому было принято решение использовать его для тестирования работоспособности бота.

Основная аудитория питомника в Instagram:

- любители животных;
- любители кошек;
- клиенты, проживающие на территории Российской Федерации.

2.2 Постановка задачи

2.2.1 Актуальность проекта

Instagram сегодня — это уже не просто приложение для обмена фотографиями и их оценки, это мощный инструмент для рекламы своего проекта. Согласно статье, на сайте www.kommersant.ru за 25.09.2018 ежемесячная аудитория этого фото-сервиса составила 800 млн. активных пользователей,

из которых 500 млн пользуются им каждый день. Кроме того, представители Instagram сообщили ресурсу Techcrunch о том, что сейчас с ним сотрудничают более 2 млн. рекламодателей, тогда как в марте их был 1 млн.

Если вы открываете свое дело, или собираетесь начать продвигать себя или свои услуги, то аккаунт в Instagram одна из первых вещей которые необходимо иметь. Вы можете спокойно выкладывать там примеры своих работ, услуг, предложений и начинать собирать вокруг своего проекта аудиторию, заинтересованную в вас.

У этой социальной сети огромная аудитория, которая каждый день пользуется этой сетью, и все что остается это просто привлекать внимание этих людей, показывая, что ты можешь им предложить, чем можешь их заинтересовать. Но и эта задача занимает много времени, особенно если человек только начинает свое дело, у него попросту нет времени тратить по несколько часов каждый день на нахождение в Instagram. Поэтому необходим инструмент, который будет автоматически выполнять поиск и оценку фотографий людей, которые могут быть заинтересованы в проекте, от имени которого ведется работа.

К сожалению, обеспечить такой функционал могут только специальные сайты или приложения для смартфонов, которые просят за свои услуги деньги. Зачастую в счет входят и различные дополнительные функции, которые многим никогда не пригождаются. Также не самым удобным решением является то, когда необходимо или заходить в браузер с телефона и настраивать все на сравнительно небольшом экране, или искать телефон с установленным приложением, когда под рукой нет компьютера. Именно в такие моменты хочется, чтобы была возможность не задумываясь переключаться между рабочим компьютером, смартфоном, планшетом или ноутбуком, без каких-либо прерываний процесса настройки или получения отчетности, притом бесплатная.

Конечно, подобные сервисы не гарантируют прирост клиентов, они лишь обеспечивают поток посетителей на ваш профиль, позволяя большему количеству людей узнать о вас или вашем проекте. И соответственно, чем точнее определена основная аудитория и чем лучше оформлен аккаунт, тем сильнее будет отдача и соответственно тем больше будет поток клиентуры.

2.2.2 Цель и назначение проекта

Цель проекта — это максимально упростить процесс управления аккаунтом в Instagram и сделать эту технологию общедоступной. Необходимо добиться того, чтобы любой человек с любым мобильным устройством, компьютером или планшетом мог пользоваться сервисом с возможностью без проблем менять рабочее устройство.

Проект должен быть реализован в виде чат-бота, позволяющего путем отправки определенных данных и команд производить управление своим Instagram аккаунтом. Минимальный функционал должен обеспечивать поиск и оценку фотографий по тегам, возможность игнорирования определенных аккаунтов и предоставлять статистику лайки/подписчики.

Взаимодействие с сервисом будет происходить через переписку с чат ботом в мессенджере Telegram, так как это приложение работает на любых мобильных устройствах, компьютерах и планшетах, и предоставляет возможность разрабатывать и использовать специальные боты.

2.2.3 Описание функционала

Для максимально эффективного пользования Telegram-ботом он должен обладать набором основным набором функций приложения Instagram и иметь возможность настраивать некоторые показатели для того, чтобы более выборочно подходить к поиску и обработке фотографий. Так же бот должен

при необходимости собирать и предоставлять статистику аккаунта, под которым выполнен вход. Должна быть правильно выстроена логика обработки запросов пользователя, во избежание вылетов или ошибочных действий со стороны бота.

Таким образом, бот должен обладать следующим набором функций:

- авторизация в социальной сети Instagram;
- поиск фотографий по заранее введенным тегам;
- оценка подходящих фотографий;
- предоставление информации о том, какие данные пользователь указал для поиска;
- настройка минимального и максимального количества оценок на найденных фотографиях;
- указание количества фотографий, которое будет обрабатываться по каждому тегу;
- возможность указать слова, при наличии которых в никнейме автора обрабатываемой фотографии эта фотография будет проигнорирована;
- предоставление статистики аккаунта, под которым выполнен вход.

2.2.4 Архитектура приложения

При отправке сообщения пользователем, оно в зашифрованном виде отсылается на сервера Telegram, откуда автоматически перенаправляется в Telegram-бот. Далее в боте происходит обработка сообщения, если оно отправлено для внесения настроек, то происходит запись новых данных и отправка ответного сообщения пользователю. Если же это сообщение направлено на начало обработки фотографий или сбор статистики, то бот отправляет соответствующий запрос на информацию в Instagram и отсылает ответ пользователю (рисунок 10).

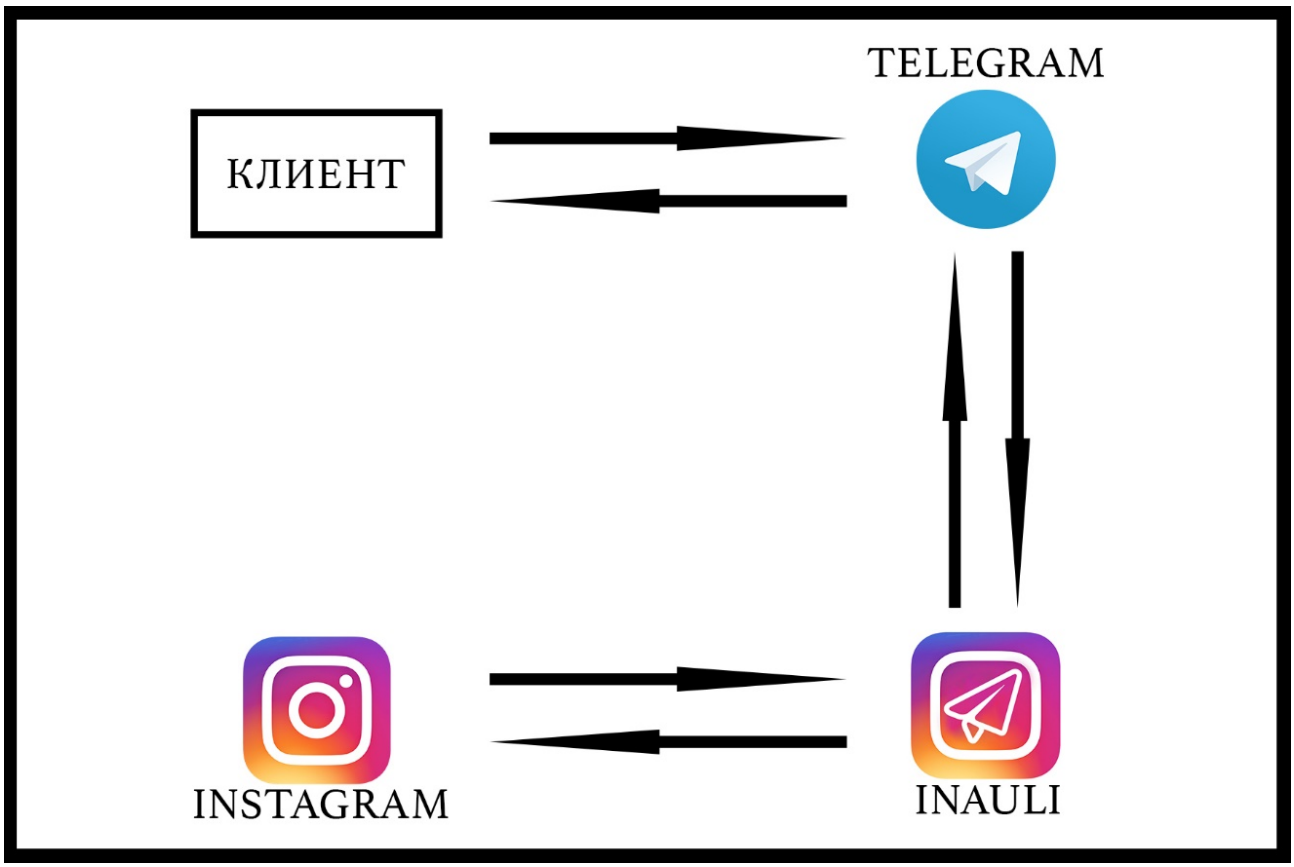


Рисунок 10 — Архитектура работы бота

2.3 Разработка Telegram-бота

2.3.1 Регистрация бота в Telegram

Для начала необходимо найти пользователя BotFather, который также является ботом, позволяющим осуществлять регистрацию и настройку пользовательских ботов (рисунок 11) [20]. BotFather позволяет присвоить имя нашему боту, задать его описание, изображение и список команд, доступных пользователям. Необходимо заполнить все пункты, так как в дальнейшем от этого будет зависеть то, насколько ботом удобно пользоваться. При регистрации выдается уникальный ключ, необходимый для связи пользователей с ботом и организации их взаимодействия.

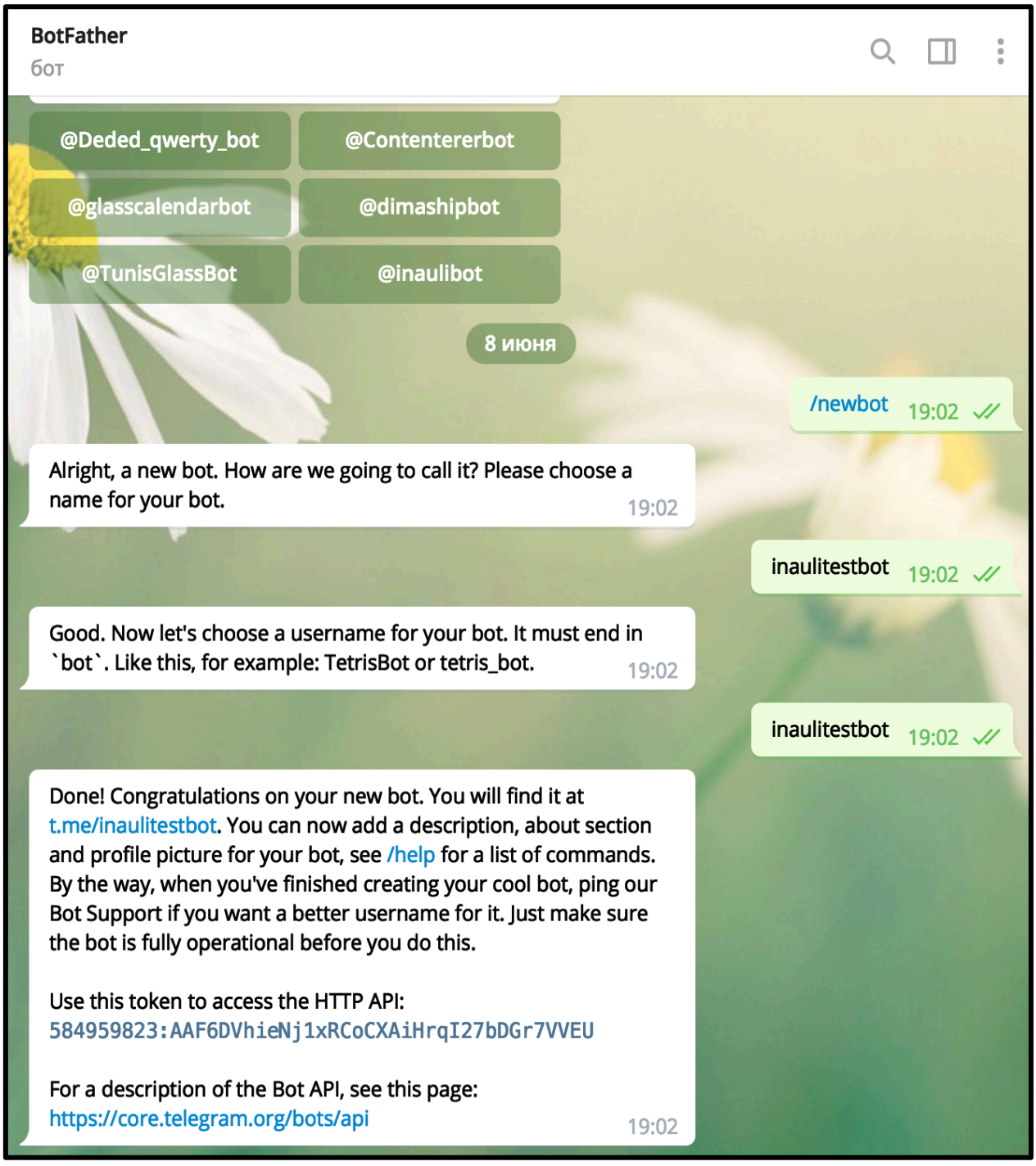


Рисунок 11 — Регистрация бота

Отправляем команду /newbot и вводим имя нашего бота и его ник, по которому его будут находить другие пользователи (рисунок 12). В дальнейшем все настройки, кроме никнейма, можно будет поменять. При необходимости также можно получить новый уникальный ключ.

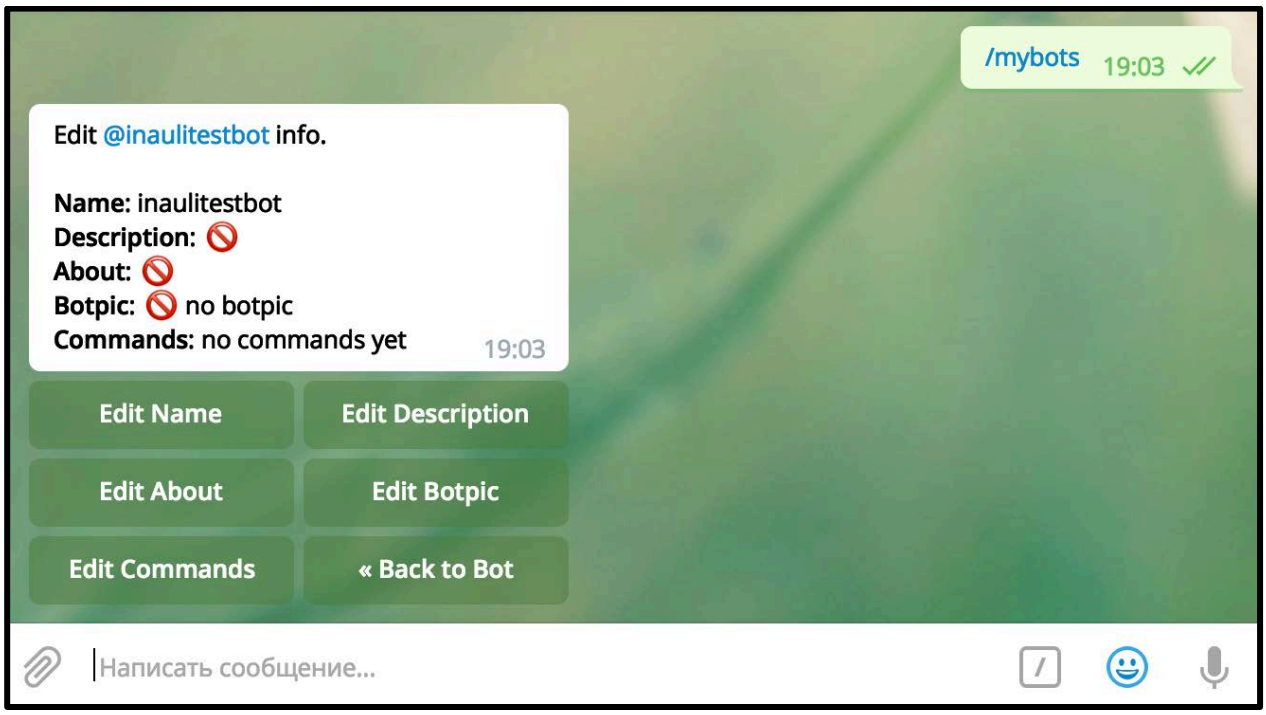


Рисунок 12 — Окно настройки бота

Далее можно добавить описание бота, которое пользователь будет получать вначале диалога, расписать команды, которые позволят ускорить ввод сообщений и присвоить боту картинку для упрощения процесса поиска.

Также BotFather дает нам уникальный ключ, при помощи которого и будет осуществляться клиент-серверная связь.

2.3.2 Добавление сторонних библиотек

Хотя множество задач можно решить стандартными средствами visual studio, иногда использование сторонних библиотек бывает не только полезным, но и необходимым. Многие библиотеки служат для простого упрощения работы, позволяя избегать сложных конструкций или упрощая синтаксис. Но в некоторых случаях, как например в нашем, сторонние библиотеки являются обязательными, так как без их наличия невозможно будет получать необходимые данные с серверов, для которых эти библиотеки и нужны.

Для полноценной работы бота необходимы библиотеки, которые обеспечат возможность обрабатывать сообщения, а также отправлять и получать информацию из Instagram.

InstaSharper — библиотека, позволяющая осуществлять вход под своим аккаунтом Instagram, поиск постов и профилей и оценивать посты пользователей от имени пользователя, под которым был осуществлен вход.

Telegram.Bot позволяет принимать и отправлять сообщения пользователей, осуществлять их обработку и предоставляет необходимую информацию о пользователях.

SysTem.Drawing.Primitives — библиотека, предоставляющая возможность создавать изображения, что необходимо для рисования графиков статистики.

2.3.3 Написание кода

Telegram-бот состоит из Program.cs, в котором находятся методы приема, обработки и отправки сообщений, и дополнительного класса Users.cs, который отвечает за работы с Instagram.

В Program.cs помимо стандартных библиотек, используются библиотеки Telegram.Bot. Они позволяют организовывать связь с пользователем и обрабатывать сообщения (рисунок 13).

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Telegram.Bot;
using Telegram.Bot.Args;
using System.IO;
using System.Security.Cryptography;
using System.Text;
using Telegram.Bot.Types.Enums;
using Telegram.Bot.Types.InputMessageContents;
using System.Net.Http;
using Telegram.Bot.Types.InlineQueryResults;
using Telegram.Bot.Types.ReplyMarkups;
using Telegram.Bot.Types.InlineKeyboardButtons;
using Telegram.Bot.Types;
using System.Net.Http.Headers;
using System.Net;
using System.Net.Configuration;
using System.Runtime.Serialization.Formatters.Binary;
```

Рисунок 13 — Список подключенных библиотек в Program.cs

Перед главным методом создается сам клиент Telegram-бота с использованием уникального ключа, полученного от BotFather, и прокси-сервера. Также создается список, в котором будут храниться пользователи, пользующиеся ботом в момент его работы и переменные, необходимые для корректной работы при запуске (рисунок 14).

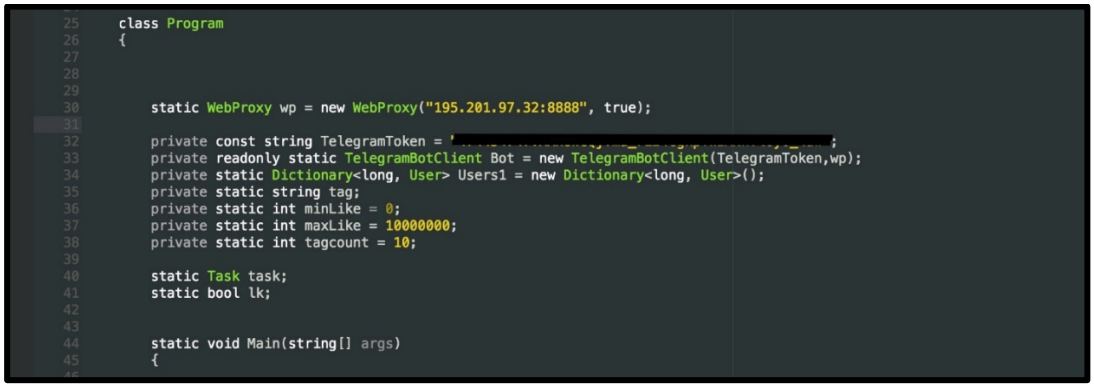


Рисунок 14 — Начало кода

В методе Main вызывается метод open загружающий основную информацию о пользователях при запуске бота. Далее подключается метод OnMessage, который будет срабатывать каждый раз, когда боту будет приходить сообщение. Затем бот отсылает мне тестовое сообщение, подтверждающее наличие стабильного подключения и отправляет в консоль сообщение «Начали», так же необходимое для подтверждения начала работы. Теперь бот будет работать до тех пор, пока администратор не нажмет на любую клавишу находясь в консоли и затем бот прекратит прием сообщений и отключится (рисунок 15).

Рисунок 15 — Содержимое метода Main

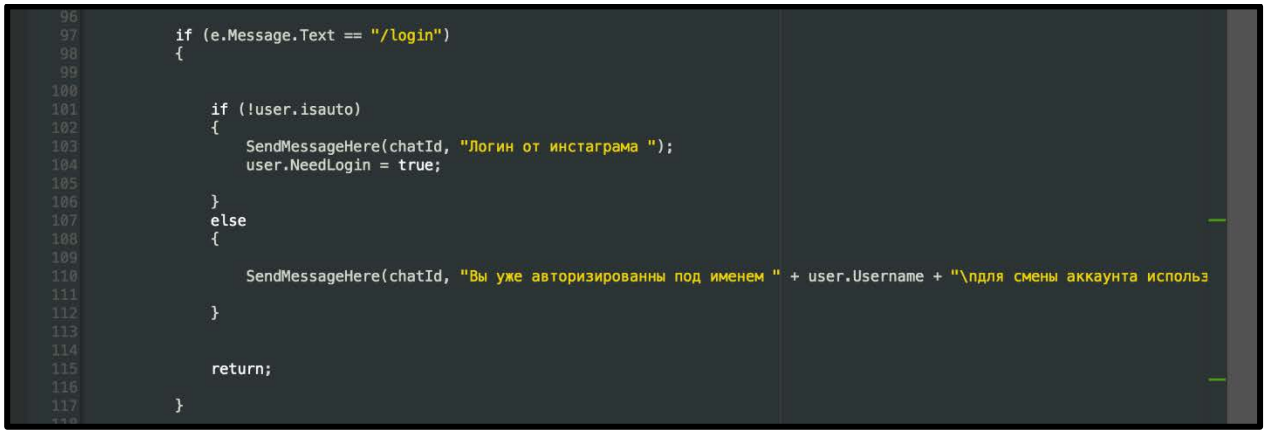
При получении сообщения начинает свою работу метод Message, который сразу узнает id пользователя, отправившего сообщение, он необходим для организации обратной связи и идентификации отправителя. После чего следует проверка на то, есть ли этот пользователь в нашем списке и можем ли мы его авторизировать. Если такого пользователя нет, то создается новый элемент класса Users и заносится в список. Ключом идентификации пользователей является их id. Если пользователь есть в списке, но не авторизирован, то происходит авторизация. И, наконец, если пользователь есть в списке и авторизирован, то метод продолжает обработку сообщения (рисунок 16).

```
private static void Message(object sender, MessageEventArgs e)
{



    var chatId = e.Message.Chat.Id;
    Console.WriteLine(chatId + ": " + e.Message.Text);
    User user = null;
    if (Users1.ContainsKey(chatId))
    {
        user = Users1[chatId];
        if (user.start)
        {
            if (user.Username != null && user.Password != null && !user.isauto)
            {
                user.Login();
                return;
            }
        }



    }
    else
    {
        Users1.Add(chatId, user = new User());
        user.ChatId = chatId;

    }
```

Рисунок 16 — Метод Message. Часть 1

Дальше начинается проверка сообщения, отправленного пользователем, на соответствие командам. Если сообщением была команда /login, то бот проверяет авторизирован ли пользователь. Если да, то бот сообщает имя аккаунта, под которым выполнен вход и предлагает воспользоваться командой

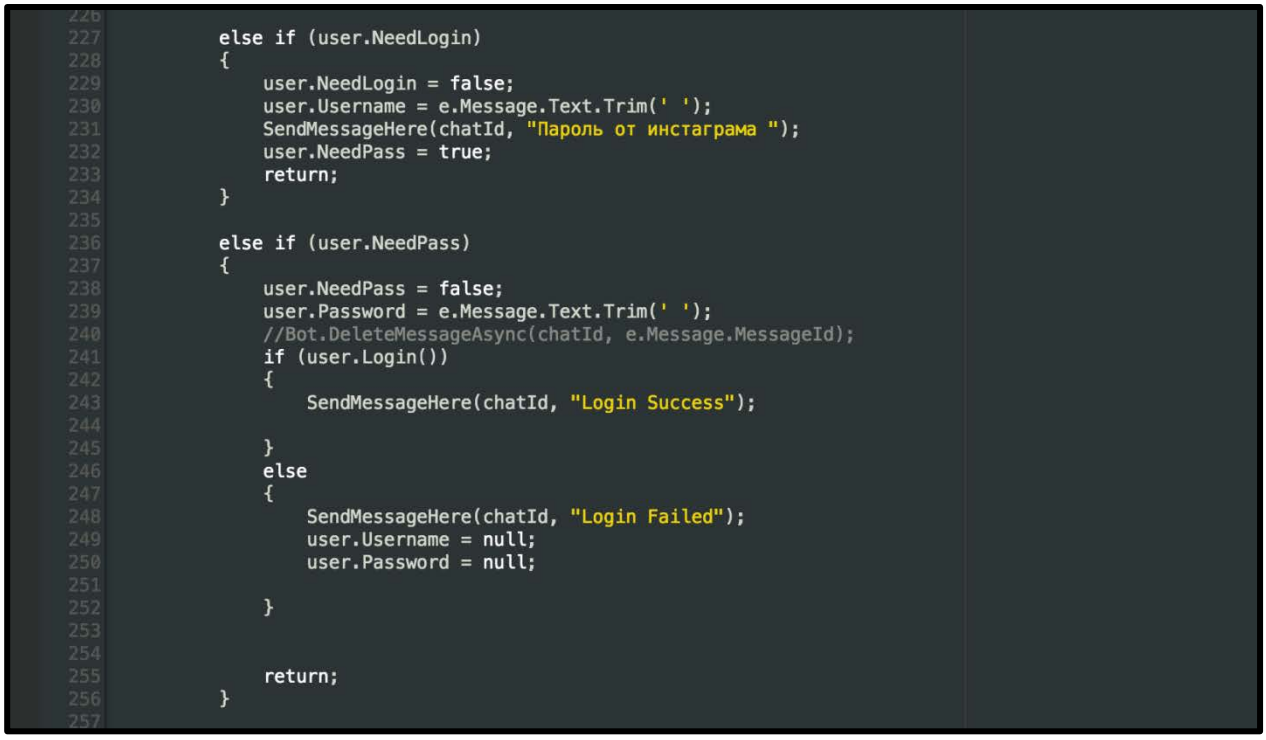
/logout для смены аккаунта. Если пользователь не авторизирован, то начинается процесс авторизации (рисунок 17).



```
96
97          if (e.Message.Text == "/login")
98          {
99
100
101             if (!user.isauto)
102             {
103                 SendMessageHere(chatId, "Логин от инстаграма ");
104                 user.NeedLogin = true;
105
106             }
107             else
108             {
109
110                 SendMessageHere(chatId, "Вы уже авторизированны под именем " + user.Username + "\nдля смены аккаунта использ
111
112             }
113
114
115             return;
116
117         }
```

Рисунок 17 — Метод Message. Часть 2

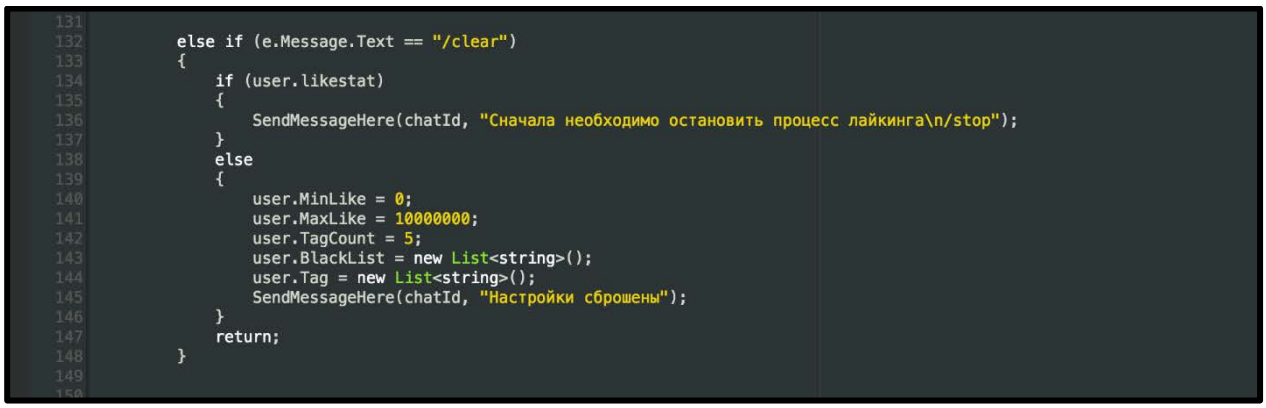
После начала авторизации бот запрашивает логин и пароль от аккаунта пользователя в Instagram и вызывает метод Login. Затем отправляется сообщение о результате попытки авторизации (рисунок 18).



```
226
227         else if (user.NeedLogin)
228         {
229             user.NeedLogin = false;
230             user.Username = e.Message.Text.Trim(' ');
231             SendMessageHere(chatId, "Пароль от инстаграма ");
232             user.NeedPass = true;
233             return;
234         }
235
236         else if (user.NeedPass)
237         {
238             user.NeedPass = false;
239             user.Password = e.Message.Text.Trim(' ');
240             //Bot.DeleteMessageAsync(chatId, e.Message.MessageId);
241             if (user.Login())
242             {
243                 SendMessageHere(chatId, "Login Success");
244
245             }
246             else
247             {
248                 SendMessageHere(chatId, "Login Failed");
249                 user.Username = null;
250                 user.Password = null;
251
252             }
253
254
255             return;
256         }
257
```

Рисунок 18 — Метод Message. Часть 3

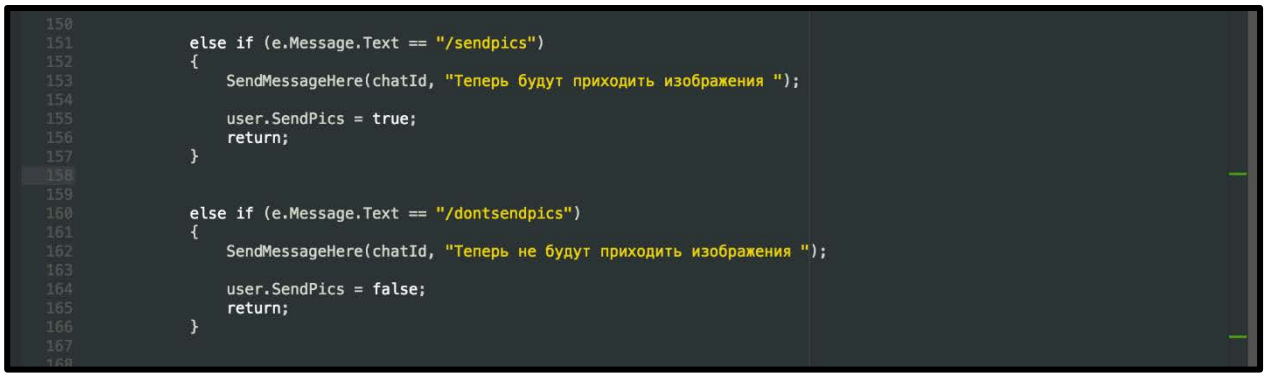
Команда /clear очищает введенные данные, возвращая настройки к стандартным. Если в данный момент бот обрабатывает изображения, то он предложит остановить процесс для выполнения очистки (рисунок 19).



```
131
132         else if (e.Message.Text == "/clear")
133         {
134             if (user.likestat)
135             {
136                 SendMessageHere(chatId, "Сначала необходимо остановить процесс лайкинга\n/stop");
137             }
138             else
139             {
140                 user.MinLike = 0;
141                 user.MaxLike = 10000000;
142                 user.TagCount = 5;
143                 user.BlackList = new List<string>();
144                 user.Tag = new List<string>();
145                 SendMessageHere(chatId, "Настройки сброшены");
146             }
147             return;
148         }
149
150
```

Рисунок 19 — Метод Message. Часть 4

По умолчанию бот не будет присылать пользователю фотографии, которые оценивает от его имени. Для управления этой функцией реализованы команды /sendpics и /dontsendpics, для включения и отключения соответственно. После отправки команды бот отсылает текущий статус функции (рисунок 20).

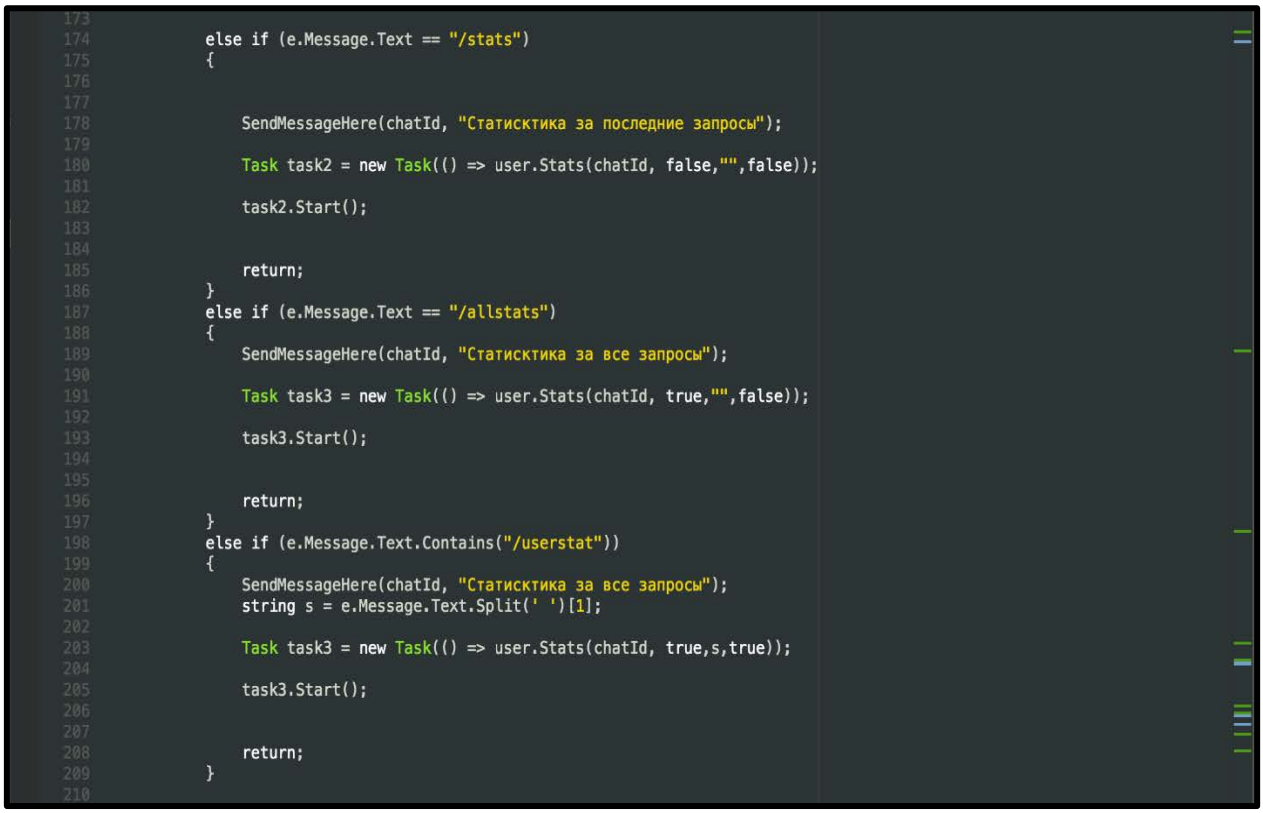


```
150
151         else if (e.Message.Text == "/sendpics")
152         {
153             SendMessageHere(chatId, "Теперь будут приходить изображения ");
154
155             user.SendPics = true;
156             return;
157         }
158
159
160         else if (e.Message.Text == "/dontsendpics")
161         {
162             SendMessageHere(chatId, "Теперь не будут приходить изображения ");
163
164             user.SendPics = false;
165             return;
166         }
167
168
```

Рисунок 20 — Метод Message. Часть 5

Для получения статистики используются следующие команды (рисунок 21):

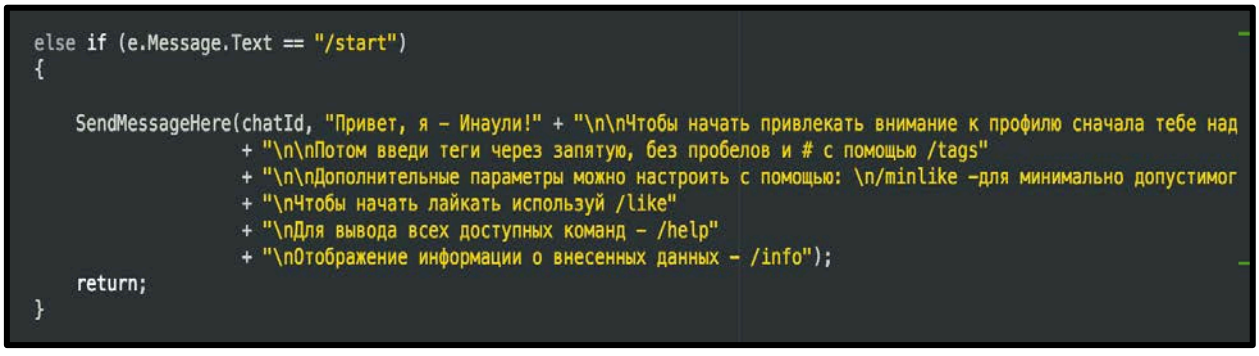
- /stats — показывает график оценок и подписчиков за последние 5 запросов в профиле аккаунта, под которым выполнен вход, и среднее количество оценок на фотографиях;
- /allstats — показывает график общего количества оценок и подписчиков в профиле аккаунта, под которым выполнен вход, и среднее количество оценок на фотографиях;
- /userstats — позволяет увидеть статистику пользователя, чей никнейм будет указан сразу после команды.



```
            else if (e.Message.Text == "/stats")
            {


                SendMessageHere(chatId, "Статисктика за последние запросы");

                Task task2 = new Task(() => user.Stats(chatId, false,"",false));

                task2.Start();


                return;
            }
            else if (e.Message.Text == "/allstats")
            {
                SendMessageHere(chatId, "Статисктика за все запросы");

                Task task3 = new Task(() => user.Stats(chatId, true,"",false));

                task3.Start();


                return;
            }
            else if (e.Message.Text.Contains("/userstat"))
            {
                SendMessageHere(chatId, "Статисктика за все запросы");
                string s = e.Message.Text.Split(' ')[1];

                Task task3 = new Task(() => user.Stats(chatId, true,s,true));

                task3.Start();


                return;
            }
```

Рисунок 21 — Метод Message. Часть 6

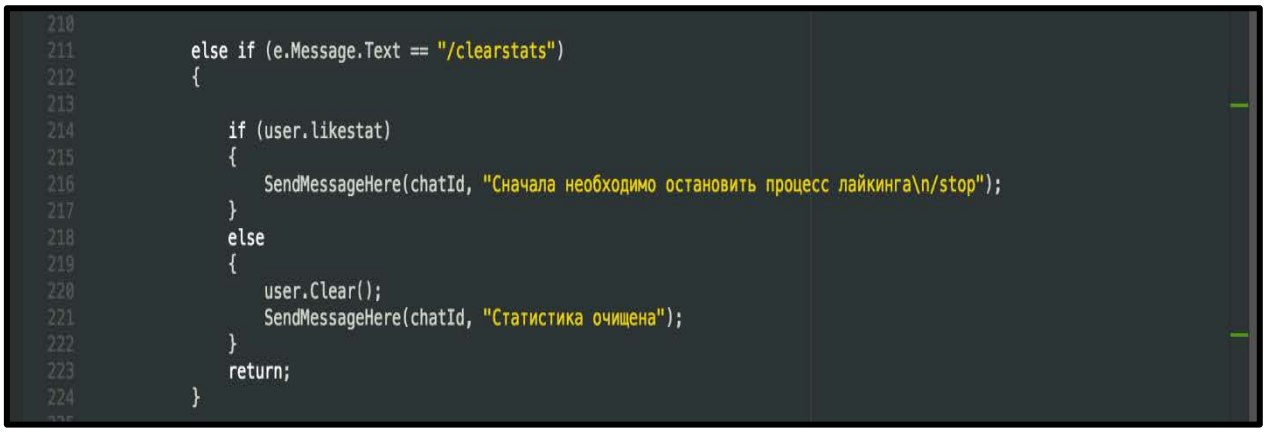
Команда /start — это стандартная команда, которая автоматически отправляется боту при первом использовании. В нашем случае ответным сообщением будет приветствие и краткая инструкция по использованию бота (рисунок 22).



```
else if (e.Message.Text == "/start")
{

    SendMessageHere(chatId, "Привет, я - Инаули!" + "\n\nЧтобы начать привлекать внимание к профилю сначала тебе над
                    + "\n\nПотом введи теги через запятую, без пробелов и # с помощью /tags"
                    + "\n\nДополнительные параметры можно настроить с помощью: \n/minlike -для минимально допустимог
                    + "\nЧтобы начать лайкать используй /like"
                    + "\nДля вывода всех доступных команд - /help"
                    + "\nОтображение информации о внесенных данных - /info");
    return;
}
```

Рисунок 22 — Метод Message. Часть 7

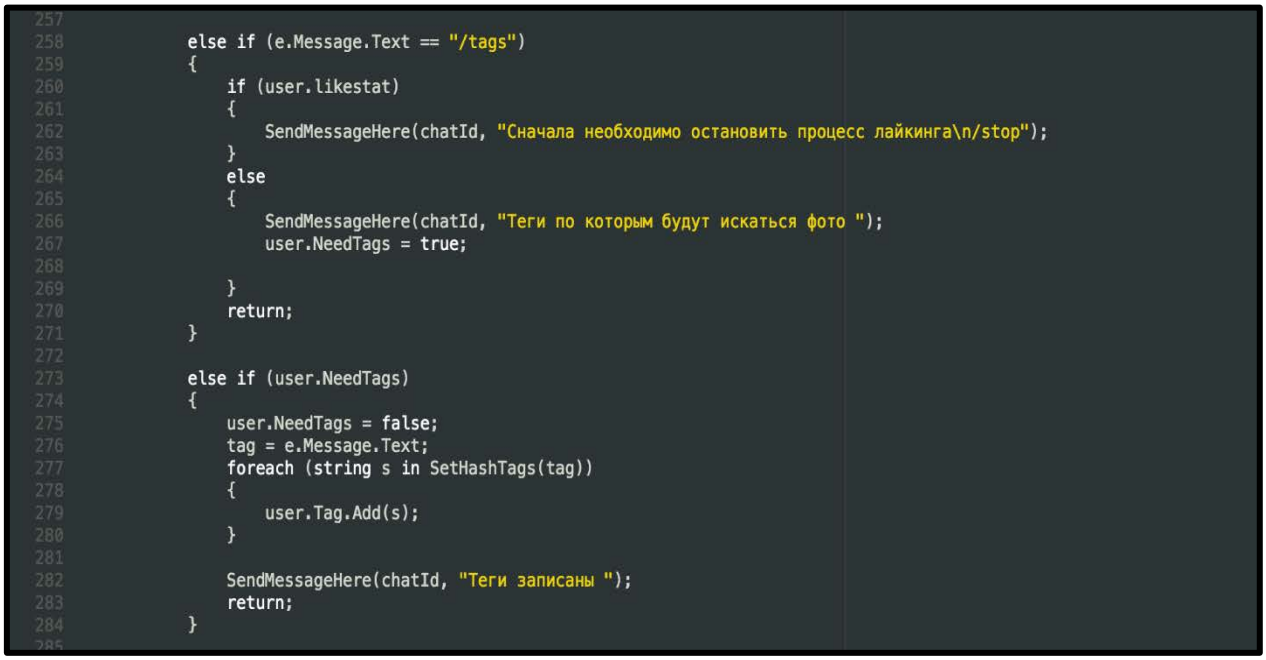
/clearstats используется в случаях, когда пользователю необходимо очистить свою историю статистики. В этом случае бот также предлагает сначала остановить процесс обработки фотографий, если он запущен (рисунок 23).



```
else if (e.Message.Text == "/clearstats")
{

    if (user.likestat)
    {
        SendMessageHere(chatId, "Сначала необходимо остановить процесс лайкинга\n/stop");
    }
    else
    {
        user.Clear();
        SendMessageHere(chatId, "Статистика очищена");
    }
    return;
}
```

Рисунок 23 — Метод Message. Часть 8

Команда /tags нужна для задания списка тегов, по которым бот будет искать фотографии в Instagram. Сообщение с тегами разбивается на отдельные слова, затем эти слова записываются в список, который в дальнейшем можно пополнить или очистить (рисунок 24).



```
else if (e.Message.Text == "/tags")
{
    if (user.likestat)
    {
        SendMessageHere(chatId, "Сначала необходимо остановить процесс лайкинга\n/stop");
    }
    else
    {
        SendMessageHere(chatId, "Теги по которым будут искаться фото ");
        user.NeedTags = true;

    }
    return;
}

else if (user.NeedTags)
{
    user.NeedTags = false;
    tag = e.Message.Text;
    foreach (string s in SetHashTags(tag))
    {
        user.Tag.Add(s);
    }

    SendMessageHere(chatId, "Теги записаны ");
    return;
}
```

Рисунок 24 — Метод Message. Часть 9

/like запускает процесс поиска и обработки фотографий или сообщает пользователю, если процесс уже запущен (рисунок 25). Метод Like запускается в отдельном потоке, что позволяет изменять некоторые настройки даже в процессе его работы.

Рисунок 25 — Метод Message. Часть 10

Команды /minlike (рисунок 26) и /maxlike (рисунок 27) используются, когда пользователю необходимо задать диапазон, при попадании в который, по количеству уже полученных оценок, фотография будет обработана.

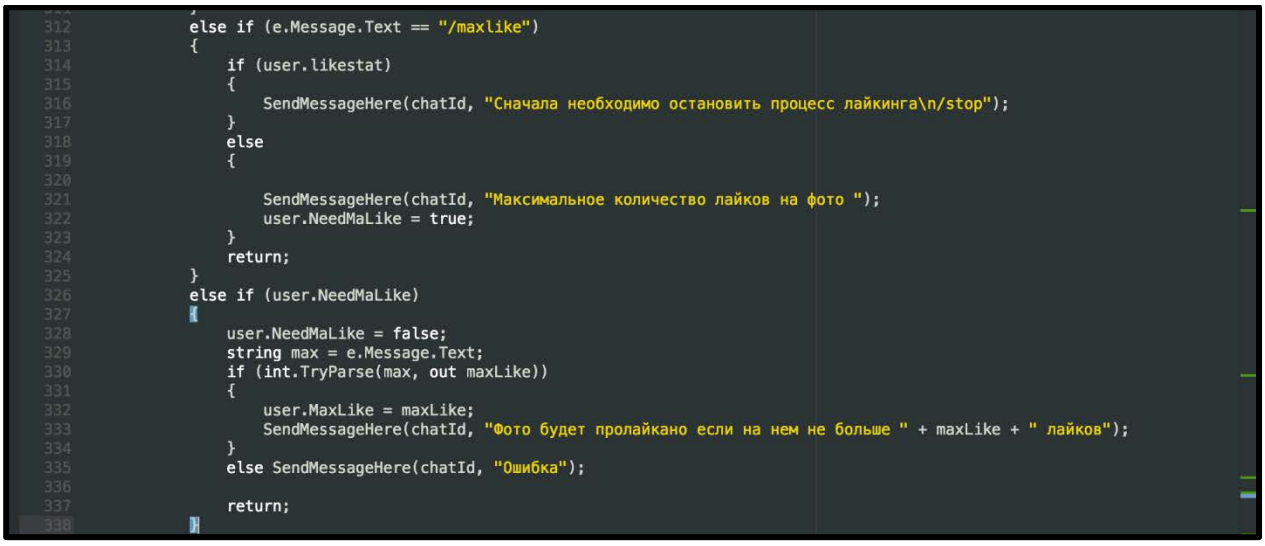


Рисунок 26 — Метод Message. Часть 11

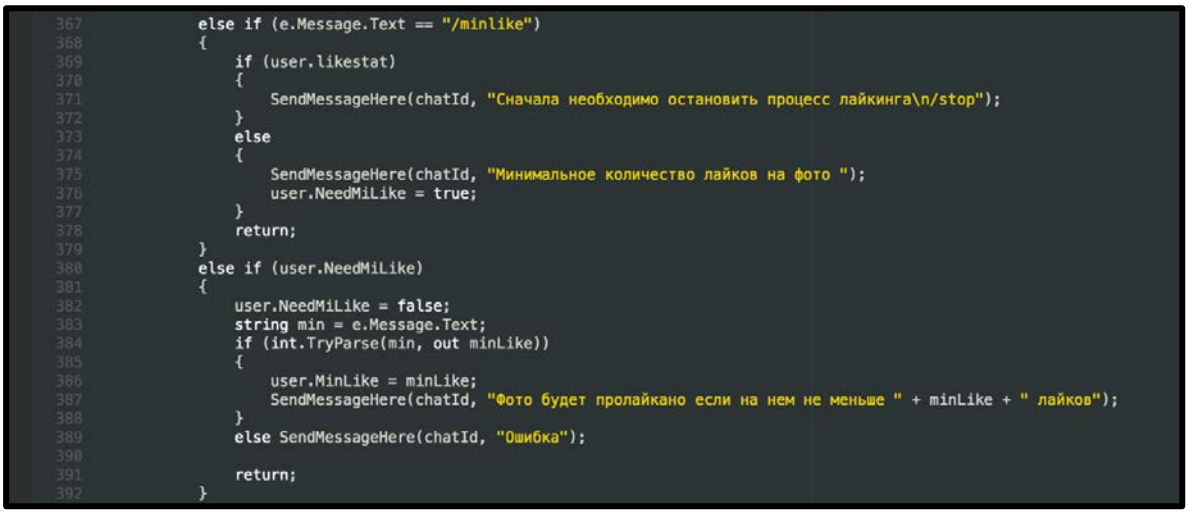


Рисунок 27 — Метод Message. Часть 12

При отправке команды /logout происходит деавторизация пользователя, и он удаляется из списка Users1 (рисунок 28).

```
            else if (e.Message.Text == "/logout")
            {


                user.Logout();
                Users1.Remove(chatId);
                logout(user);

            }
```

Рисунок 28 — Метод Message. Часть 13

При необходимости добавить определенные слова в черный список имен, отправляется команда /blacklist. В случае запущенного процесса обработки фотографий бот попросит остановить его (рисунок 29).

```
    else if (e.Message.Text == "/blacklist")

    {
        if (user.likestat)
        {
            SendMessageHere(chatId, "Сначала необходимо остановить процесс лайкинга\n/stop");
        }
        else
        {
            SendMessageHere(chatId, "введите слова, при наличии которых в имени пользователя его фото не будет пролайкано ");
            user.NeedBlackList = true;
        }
        return;
    }
    else if (user.NeedBlackList)
    {
        user.NeedBlackList = false;

        tag = e.Message.Text;
        foreach (string s in SetHashTags(tag))
        {
            user.BlackList.Add(s);
        }
        // user.BlackList = SetHashTags(tag);
        SendMessageHere(chatId, "Черный список записан");
        return;


    }
```

Рисунок 29 — Метод Message. Часть 14

Чтобы увидеть внесенные данные, используется команда /info, вызывающая метод Info. Этот метод отправляет пользователю информацию об имени аккаунта, под которым выполнен вход, внесенных настройках, списке тегов и черном списке, отсылаются ли обработанные фотографии и имеется ли у пользователя временная блокировка в результате превышения ограничения количества обработанных фотографий (рисунок 30).

```
else if (e.Message.Text == "/info")
{

    user.Info();

    return;
}
```

Рисунок 30 — Метод Message. Часть 15

Команды /cleartag и /clearbl используются для очистки списка тегов и черного списка соответственно (рисунок 31).

```
428         else if (e.Message.Text == "/clearbl")
429         {
430             if (user.likestat)
431             {
432                 SendMessageHere(chatId, "Сначала необходимо остановить процесс лайкинга\n/stop");
433             }
434             else
435             {
436                 user.ClearBL();
437             }
438
439             return;
440         }
441         else if (e.Message.Text == "/cleartag")
442         {
443             if (user.likestat)
444             {
445                 SendMessageHere(chatId, "Сначала необходимо остановить процесс лайкинга\n/stop");
446             }
447             else
448             {
449                 user.ClearTag();
450             }
451
452             return;
453         }
```

Рисунок 31 — Метод Message. Часть 16

При необходимости прервать процесс обработки фотографий используется команда /stop (рисунок 32).

```
454         else if (e.Message.Text == "/stop")
455         {
456             if (!task.IsCompleted)
457             {
458                 user.stop = true;
459             }
460             return;
461         }
```

Рисунок 32 — Метод Message. Часть 17

Команда /help возвращает пользователю список всех доступных команд с их описанием (рисунок 33).

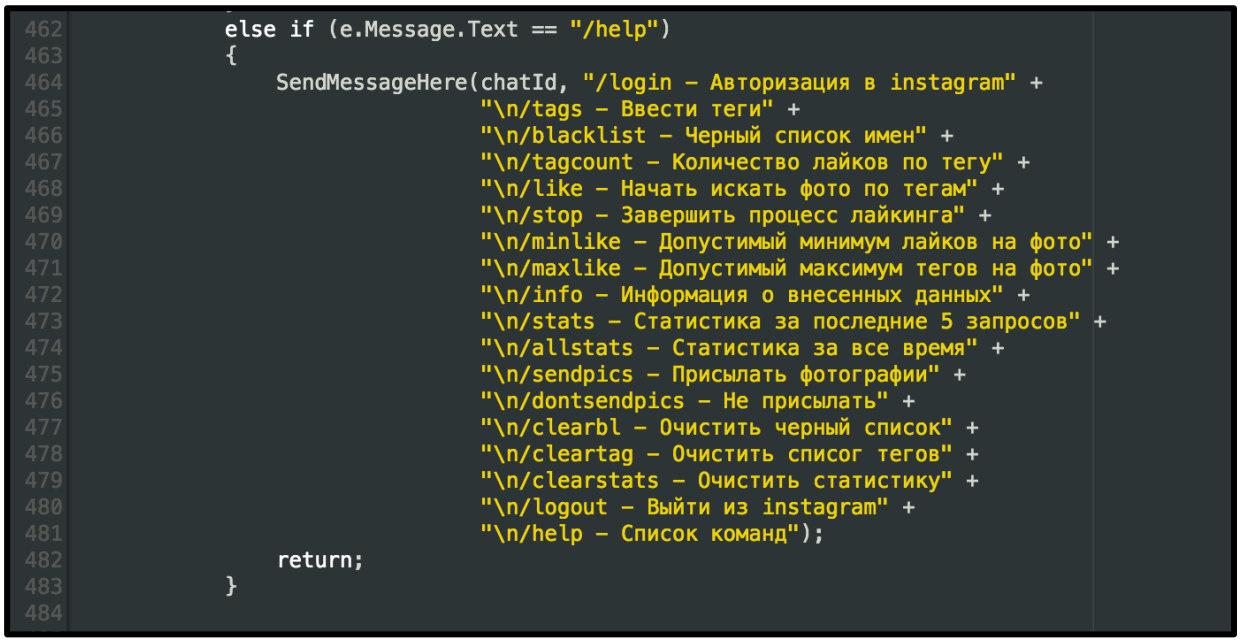


Рисунок 33 — Метод Message. Часть 18

При отсутствии совпадений бот отправляет соответствующее сообщение (рисунок 34).

Рисунок 34 — Метод Message. Часть 19

SetHashTags выполняет функцию разделения сообщения пользователя на отдельные слова и удаления из них лишних элементов. Это необходимо для записи в список тегов или черный список (рисунок 35).

Рисунок 35 — Метод SetHashTags

SendMessageHere и SendMessage — это статичный и не статичный методы, необходимые для отправки пользователю сообщений из Program.cs и Users.cs соответственно (рисунок 36).

```
510         public void SendMessage(long chatId, string txt)
511         {
512             Bot.SendTextMessageAsync(chatId, txt);
513
514         }
515
516         public static void SendMessageHere(long chatId, string txt)
517         {
518             Bot.SendTextMessageAsync(chatId, txt);
519
520         }
521
```

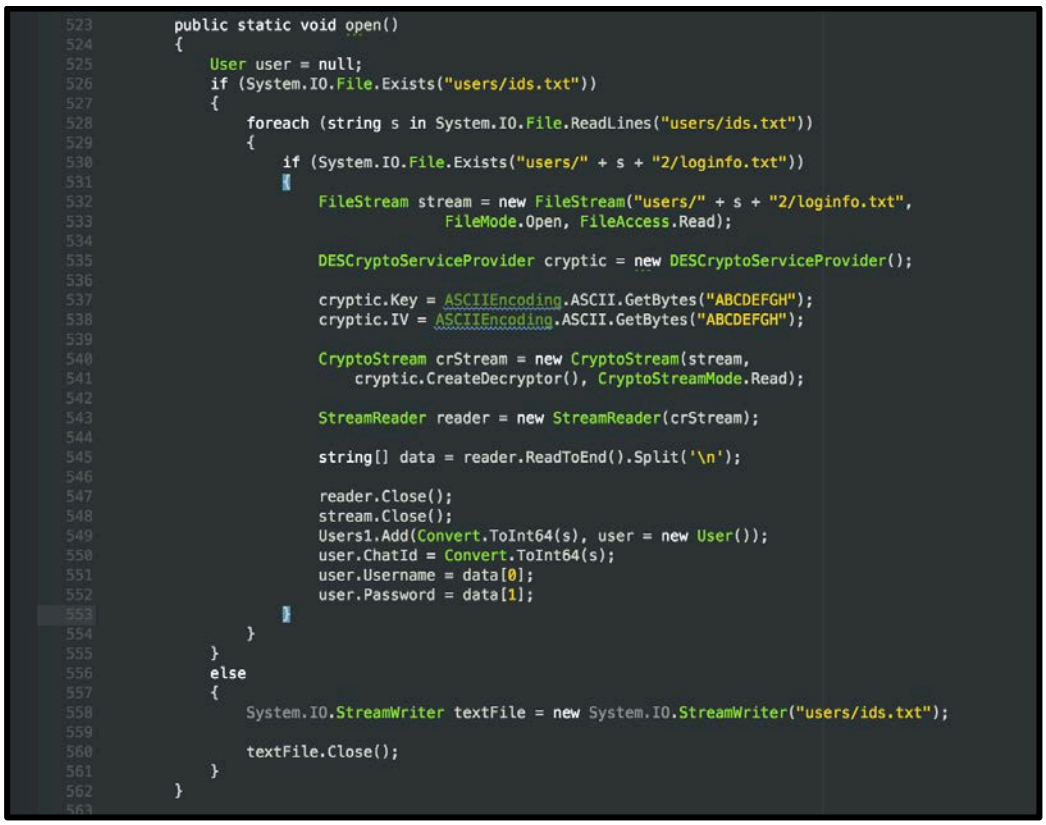
Рисунок 36 — Методы SendMessage и SendMessageHere

Метод logout запускает процесс деавторизации пользователя, при необходимости смены рабочего аккаунта (рисунок 37).

```
600
601         public static void logout(User us)
602         {
603             if (System.IO.File.Exists("users/ids.txt"))
604             {
605                 foreach (string s in System.IO.File.ReadAllLines("users/ids.txt"))
606                 {
607                     if (s == us.ChatId.ToString())
608                     {
609                         string tempFile = Path.GetTempFileName();
610
611                         using (var sr = new StreamReader("users/ids.txt"))
612                         using (var sw = new StreamWriter(tempFile))
613                         {
614                             string line;
615
616                             while ((line = sr.ReadLine()) != null)
617                             {
618                                 if (line != s)
619                                     sw.WriteLine(line);
620                             }
621                         }
622
623                         System.IO.File.Delete("users/ids.txt");
624                         System.IO.File.Move(tempFile, "users/ids.txt");
625                     }
626                 }
627             }
628             if (System.IO.File.Exists("users/" + us.ChatId + "2/loginfo.txt"))
629             {
630                 System.IO.File.Delete("users/" + us.ChatId + "2/loginfo.txt");
631             }
632         }
633
```

Рисунок 37 — Метод logout

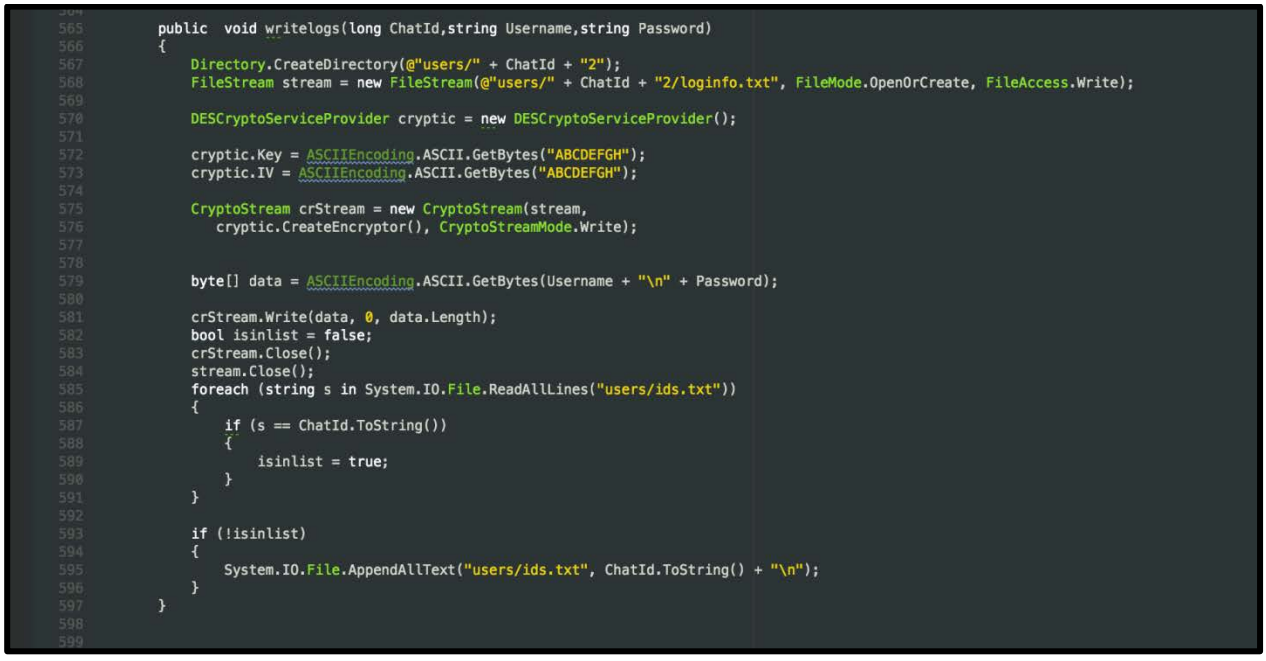
Метод open срабатывает при запуске бота, он загружает и расшифровывает сохраненную информацию о пользователях (рисунок 38).



```
523         public static void open()
524         {
525             User user = null;
526             if (System.IO.File.Exists("users/ids.txt"))
527             {
528                 foreach (string s in System.IO.File.ReadLines("users/ids.txt"))
529                 {
530                     if (System.IO.File.Exists("users/" + s + "2/loginfo.txt"))
531                     {
532                         FileStream stream = new FileStream("users/" + s + "2/loginfo.txt",
533                                     FileMode.Open, FileAccess.Read);
534 
535                         DESCryptoServiceProvider cryptic = new DESCryptoServiceProvider();
536 
537                         cryptic.Key = ASCIIEncoding.ASCII.GetBytes("ABCDEFGH");
538                         cryptic.IV = ASCIIEncoding.ASCII.GetBytes("ABCDEFGH");
539 
540                         CryptoStream crStream = new CryptoStream(stream,
541                             cryptic.CreateDecryptor(), CryptoStreamMode.Read);
542 
543                         StreamReader reader = new StreamReader(crStream);
544 
545                         string[] data = reader.ReadToEnd().Split('\n');
546 
547                         reader.Close();
548                         stream.Close();
549                         Users1.Add(Convert.ToInt64(s), user = new User());
550                         user.ChatId = Convert.ToInt64(s);
551                         user.Username = data[0];
552                         user.Password = data[1];
553                     }
554                 }
555             }
556             else
557             {
558                 System.IO.StreamWriter textFile = new System.IO.StreamWriter("users/ids.txt");
559 
560                 textFile.Close();
561             }
562         }
563 
```

Рисунок 38 — Метод Open

WriTelogs используется при обновлении пользователем информации о себе, эта информация шифруется и записывается в соответствующий файл (рисунок 39).



```
565         public  void writelogs(long ChatId,string Username,string Password)
566         {
567             Directory.CreateDirectory(@"users/" + ChatId + "2");
568             FileStream stream = new FileStream(@"users/" + ChatId + "2/loginfo.txt", FileMode.OpenOrCreate, FileAccess.Write);
569 
570             DESCryptoServiceProvider cryptic = new DESCryptoServiceProvider();
571 
572             cryptic.Key = ASCIIEncoding.ASCII.GetBytes("ABCDEFGH");
573             cryptic.IV = ASCIIEncoding.ASCII.GetBytes("ABCDEFGH");
574 
575             CryptoStream crStream = new CryptoStream(stream,
576                cryptic.CreateEncryptor(), CryptoStreamMode.Write);
577 
578 
579             byte[] data = ASCIIEncoding.ASCII.GetBytes(Username + "\n" + Password);
580 
581             crStream.Write(data, 0, data.Length);
582             bool isinlist = false;
583             crStream.Close();
584             stream.Close();
585             foreach (string s in System.IO.File.ReadAllLines("users/ids.txt"))
586             {
587                 if (s == ChatId.ToString())
588                 {
589                     isinlist = true;
590                 }
591             }
592 
593             if (!isinlist)
594             {
595                 System.IO.File.AppendAllText("users/ids.txt", ChatId.ToString() + "\n");
596             }
597         }
598 
599 
```

Рисунок 39 — Метод wriTelogs

Это все методы, использующиеся в Program.cs и отвечающие за взаимодействия пользователя с ботом. За взаимодействие бота с Instagram отвечает Users.cs. Разбор кода Users.cs приведен ниже.

Библиотеки, используемые в Users.cs (рисунок 40).

```
using System;
using System.Collections.Generic;
using System.Threading;
using InstaSharper.API;
using InstaSharper.API.Builder;
using InstaSharper.Classes;
using System.Drawing;
using System.Net.Http;
using System.Threading.Tasks;
using System.Text;
using System.IO;
using System.Linq;
using System.Timers;
using System.Net;
```

Рисунок 40 — Используемые библиотеки в User.cs

Сразу создается prog, для использования метода SendMessage. Затем объявляется переменная _instaApi, с помощью которой мы будем отправлять и получать информацию в Instagram. После этого создаются списки и счетчики, которые послужат для сбора статистики. И в конце идет таймер, который запустится при превышении лимита обработки фотографий (рисунок 41).

```
static Program prog = new Program();
private  IInstaApi _instaApi;
List<double> likes = new List<double>();
List<string> dates = new List<string>();
List<double> likes5 = new List<double>();
List<double> foll5 = new List<double>();
List<double> foll = new List<double>();
public int ik2 = 0;
bool first = true;
private const string TelegramToken = "474454747:AAH8WCQjVma_ri24cgkpTWBMNhvlc

[NonSerialized] System.Timers.Timer timer = new System.Timers.Timer(7200000);
```

Рисунок 41 — Начало User.cs

Далее идет конструктор User, который сразу задает новому пользователю стандартные значения (рисунок 42).

```
        public User()
        {
            MinLike = 0;
            MaxLike = 10000000;
            TagCount = 5;
            BlackList = new List<string>();
            Tag = new List<string>();
            likestat = false;
            stop = false;
            start = true;
            isauto = false;

            return;


        }
```

Рисунок 42 — Конструктор User

Затем следует описание свойств, которые будут содержать в себе всю необходимую информацию о каждом пользователе (рисунок 43).

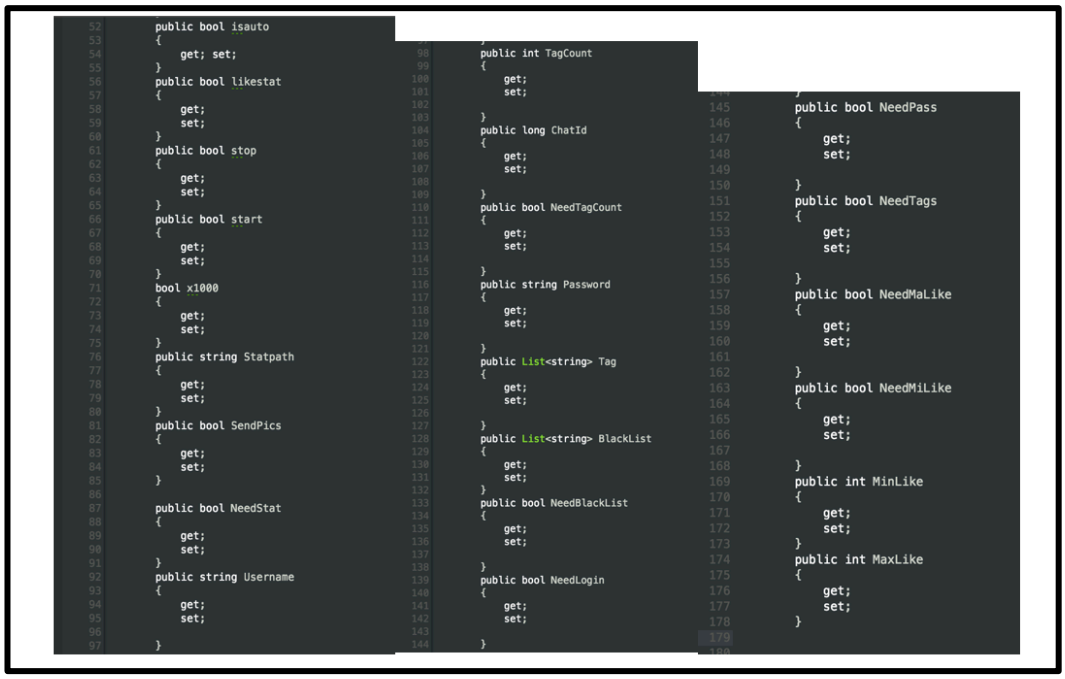
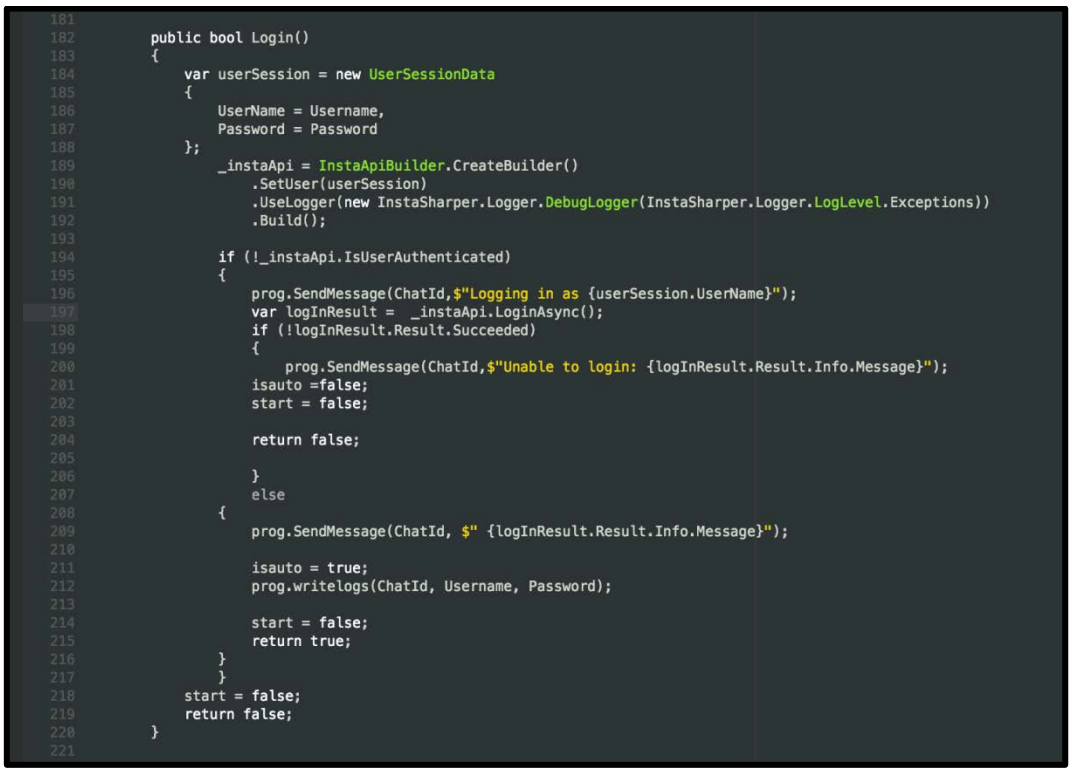


Рисунок 43 — Используемые свойства

Метод Login отвечает за авторизацию пользователя в Instagram. Если не удалось выполнить вход, то бот уведомляет об этом и присылает текст сообщения об ошибке, чтобы можно было узнать причину неудачи (рисунок 44).

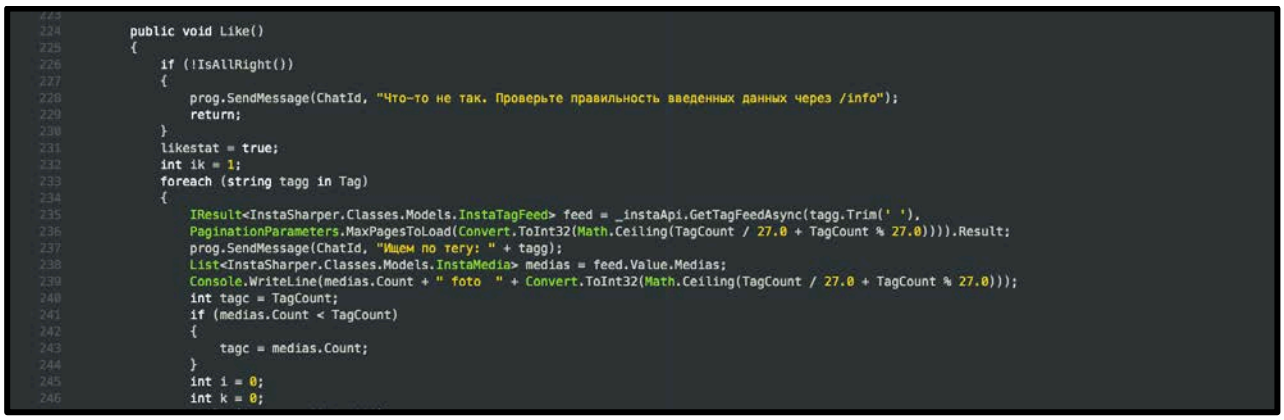


```
181
182         public bool Login()
183         {
184             var userSession = new UserSessionData
185             {
186                 UserName = Username,
187                 Password = Password
188             };
189                 _instaApi = InstaApiBuilder.CreateBuilder()
190                     .SetUser(userSession)
191                     .UseLogger(new InstaSharper.Logger.DebugLogger(InstaSharper.Logger.LogLevel.Exceptions))
192                     .Build();
193
194                 if (!_instaApi.IsUserAuthenticated)
195                 {
196                     prog.SendMessage(ChatId,$"Logging in as {userSession.UserName}");
197                     var logInResult =  _instaApi.LoginAsync();
198                     if (!logInResult.Result.Succeeded)
199                     {
200                         prog.SendMessage(ChatId,$"Unable to login: {logInResult.Result.Info.Message}");
201                     isauto =false;
202                     start = false;
203
204                     return false;
205
206                     }
207                     else
208                 {
209                     prog.SendMessage(ChatId, $" {logInResult.Result.Info.Message}");
210
211                     isauto = true;
212                     prog.writelogs(ChatId, Username, Password);
213
214                     start = false;
215                     return true;
216                 }
217                 }
218             start = false;
219             return false;
220         }
221
```

Рисунок 44 — Метод Login

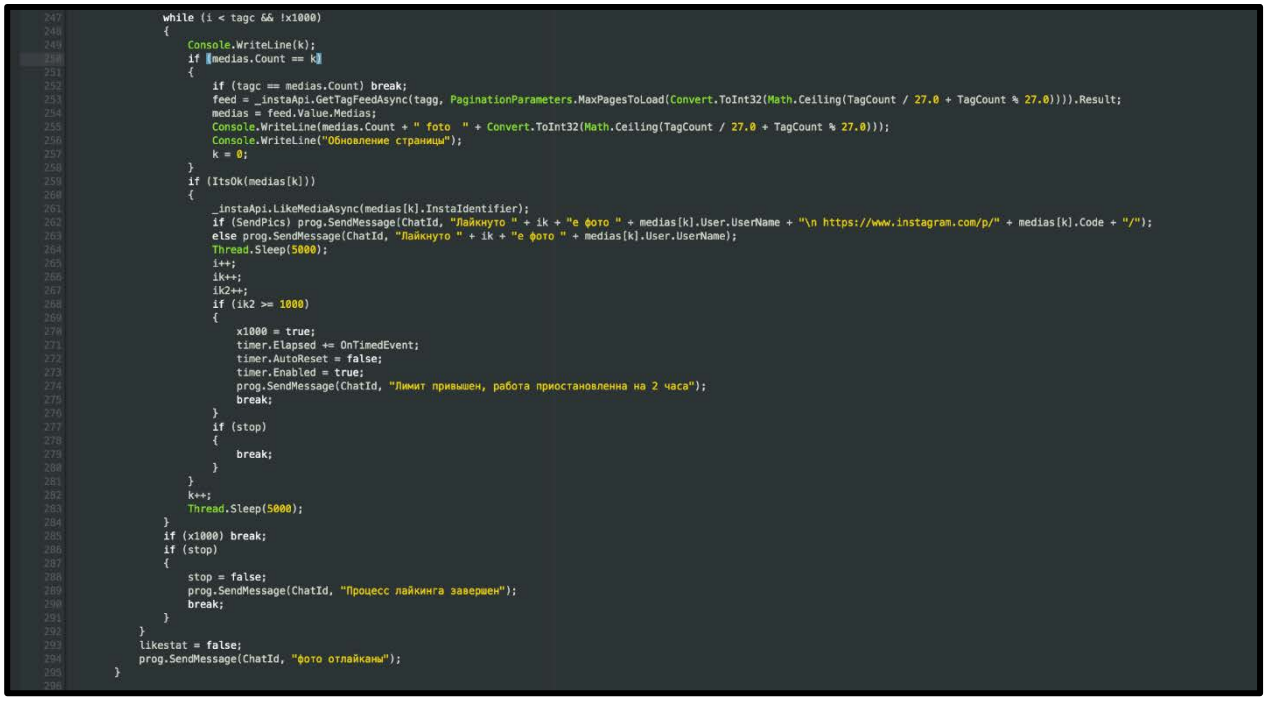
Метод Like — самый главный метод в User.cs (рисунок 45), отвечающий за поиск фотографий, загрузку и обработку информации. В нем по очереди по каждому тегу, с помощью авторской формулы, загружается информация об определенном количестве фотографий и проверяется с помощью вспомогательных методов на соответствие всем критериям отбора (рисунок 46). Если при проверке не было достигнуто необходимое количество фотографий, загружается следующая партия информации. Затем бот переходит к следующему тегу. И так продолжается до тех пор, пока поставленная задача не будет выполнена, пока не будет достигнут установленный лимит на 1000 оценок в сутки или пока процесс не будет специально прерван командой /stop. При превышении лимита запускается таймер на 3 часа, до истечения которого нельзя будет использовать метод Like. Данное ограничение введено специально для избежания подозрений и возможных блокировок со стороны Instagram.

Если были проверены все загруженные фотографии, но не набралось нужного количества, бот сообщает об этом в консоль, после чего загружает новые фотографии.



```
public void Like()
{
    if (!IsAllRight())
    {
        prog.SendMessage(ChatId, "Что-то не так. Проверьте правильность введенных данных через /info");
        return;
    }
    likestat = true;
    int ik = 1;
    foreach (string tagg in Tag)
    {
        IResult<InstaSharper.Classes.Models.InstaTagFeed> feed = _instaApi.GetTagFeedAsync(tagg.Trim(' '),
        PaginationParameters.MaxPagesToLoad(Convert.ToInt32(Math.Ceiling(TagCount / 27.0 + TagCount % 27.0)))).Result;
        prog.SendMessage(ChatId, "Ищем по тегу: " + tagg);
        List<InstaSharper.Classes.Models.InstaMedia> medias = feed.Value.Medias;
        Console.WriteLine(medias.Count + " foto  " + Convert.ToInt32(Math.Ceiling(TagCount / 27.0 + TagCount % 27.0)));
        int tagc = TagCount;
        if (medias.Count < TagCount)
        {
            tagc = medias.Count;
        }
        int i = 0;
        int k = 0;
```

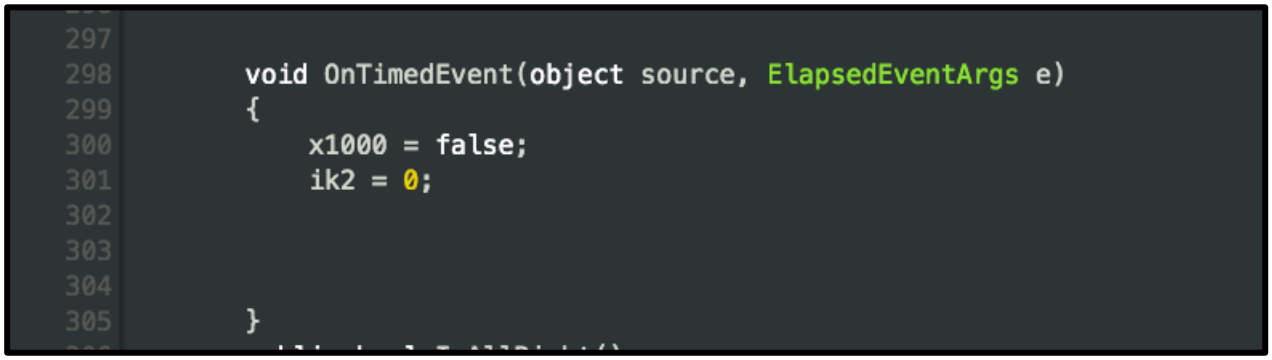
Рисунок 45 — Метод Like. Часть 1



```
        while (i < tagc && !x1000)
        {
            Console.WriteLine(k);
            if (medias.Count == k)
            {
                if (tagc == medias.Count) break;
                feed = _instaApi.GetTagFeedAsync(tagg, PaginationParameters.MaxPagesToLoad(Convert.ToInt32(Math.Ceiling(TagCount / 27.0 + TagCount % 27.0)))).Result;
                medias = feed.Value.Medias;
                Console.WriteLine(medias.Count + " foto  " + Convert.ToInt32(Math.Ceiling(TagCount / 27.0 + TagCount % 27.0)));
                Console.WriteLine("Обновление страницы");
                k = 0;
            }
            if (ItsOk(medias[k]))
            {
                _instaApi.LikeMediaAsync(medias[k].InstaIdentifier);
                if (SendPics) prog.SendMessage(ChatId, "Лайкнуто " + ik + "е фото " + medias[k].User.UserName + "\n https://www.instagram.com/p/" + medias[k].Code + "/");
                else prog.SendMessage(ChatId, "Лайкнуто " + ik + "е фото " + medias[k].User.UserName);
                Thread.Sleep(5000);
                i++;
                ik++;
                ik2++;
                if (ik2 >= 1000)
                {
                    x1000 = true;
                    timer.Elapsed += OnTimedEvent;
                    timer.AutoReset = false;
                    timer.Enabled = true;
                    prog.SendMessage(ChatId, "Лимит привышен, работа приостановленна на 2 часа");
                    break;
                }
                if (stop)
                {
                    break;
                }
            }
            k++;
            Thread.Sleep(5000);
        }
        if (x1000) break;
        if (stop)
        {
            stop = false;
            prog.SendMessage(ChatId, "Процесс лайкинга завершен");
            break;
        }
    }
    likestat = false;
    prog.SendMessage(ChatId, "фото отлайканы");
}
```

Рисунок 46 — Метод Like. Часть 2

По истечении времени блокировки присваивается значение false соответствующему свойству и обнуляется счетчик обработанных фотографий (рисунок 47).



```
297
298         void OnTimedEvent(object source, ElapsedEventArgs e)
299         {
300             x1000 = false;
301             ik2 = 0;
302
303
304
305         }
```

Рисунок 47 — Метод OnTimedEvent

Методы IsAllRight и IsOk (рисунок 48) проверяют достаточно ли информации для обработки фотографий и соответствует ли конкретная фотография всем критериям соответственно. IsAllRight проверяет авторизирован ли пользователь, записаны ли теги, указано ли количество фотографий и нет ли блокировки. ItsOk проверяет каждое фото по следующим критериям:

- входит ли количество оценок на фото в заданный диапазон;
- стоит ли оценка от аккаунта, от имени которого ведется работа;
- содержит ли никнейм владельца фотографии слова, входящие в черный список.

```
306         public bool IsAllRight()
307         {
308             return !string.IsNullOrEmpty(Username) &&
309                           Tag != null &&
310                           Tag.Count > 0 &&
311                           !x1000;
312
313
314         }
315
316         private bool ItsOk(InstaSharper.Classes.Models.InstaMedia media)
317         {
318             return media.LikesCount >= MinLike &&
319                         media.LikesCount <= MaxLike &&
320                         !media.HasLiked &&
321                         IsInBList(media.User.UserName);
322
323         }
```

Рисунок 48 — Методы IsAllRight и ItsOk

IsInBList — вспомогательный метод для ItsOk, проверяющий имя владельца фотографии (рисунок 49).

```
325         private bool IsInBList(string Name)
326         {
327             foreach (string s in BlackList)
328             {
329                 if (Name.Contains(s))
330                     return false;
331             }
332             return true;
333         }
334
```

Рисунок 49 — Метод IsInBList

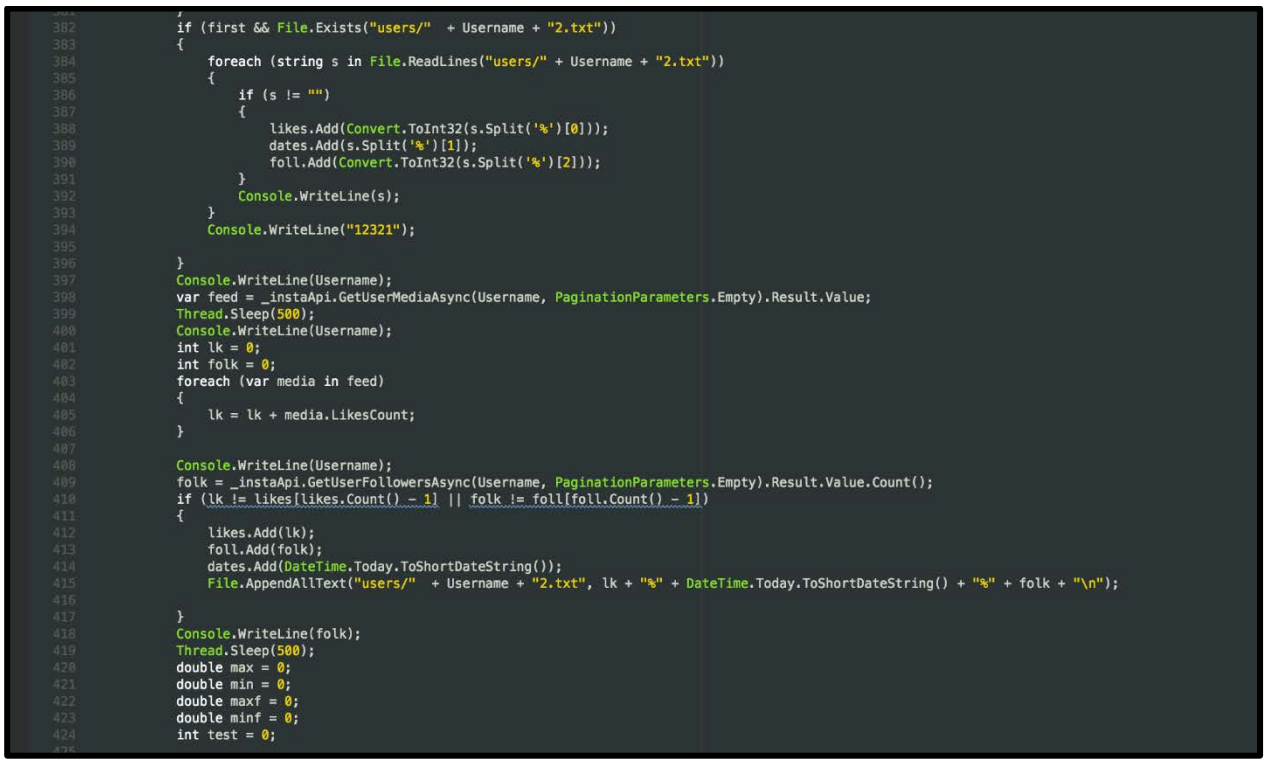
Метод Stats — большой метод, отвечающий за сбор, сохранение и загрузку статистики пользователя, создание и отправку графика.

Начинается метод с проверки авторизации пользователя и отправки соответствующего сообщения если авторизация не была произведена. Если же все в порядке, следует проверка на то, чью именно статистику запросил пользователь. Если пользователь запросил свою статистику, то далее будет использоваться его никнейм для запросов. Если же был запрос на статистику другого пользователя, то его никнейм отправляется в Instagram, для получения этого же никнейма, но с правильным регистром, так как в последующих запросах это будет важно. Если аккаунт пользователя закрытый — бот отправляет сообщение о невозможности выполнения операции (рисунок 50).

После определения никнейма бот проверяет существует ли в базе предыдущая статистика по этому пользователю, и если такая имеется, то загружает ее. Бот отправляет запрос на информацию фотографиях и количестве подписчиков пользователя в данный момент и также вносит полученные данные в список и сохраняет в базу (рисунок 51).

```
337
338     public void Stats(long chatid, bool more,string another,bool anoth)
339     {
340
341         if (!isauto)
342         {
343             prog.SendMessage(chatid, "Необходимо авторизироваться");
344             return;
345         }
346         Username = _instaApi.GetCurrentUserAsync().Result.Value.UserName;
347         if (anoth)
348         {
349             if (another != "" && !_instaApi.GetUserAsync(_instaApi.GetUserAsync(another).Result.Value.UserName).Result.Value.IsPrivate)
350             {
351                 Username =
352                     _instaApi.GetUserAsync(another).Result.Value.UserName;
353             }
354             else
355             {
356                 prog.SendMessage(ChatId, "Не получилось");
357                 return;
358
359             }
360         }
361         |
362         likes.Clear();
363         foll.Clear();
364         dates.Clear();
365         List<double> pix = new List<double>();
366         List<string> dat = new List<string>();
367         List<double> follk = new List<double>();
368         List<double> fo11 = new List<double>();
369         List<double> lik = new List<double>();
370         if (!File.Exists("users/"  + Username + "2.txt"))
371         {
372             Directory.CreateDirectory(@"users/" + ChatId + "2");
373             System.IO.StreamWriter textFile = new System.IO.StreamWriter("users/"  + Username + "2.txt");
374             textFile.Write("0%" + DateTime.Today.ToShortDateString() + "%0" + "\n");
375             textFile.Close();
376             Thread.Sleep(500);
377             likes.Add(0);
378             foll.Add(0);
379             dates.Add(DateTime.Today.ToShortDateString());
380
381         }
```

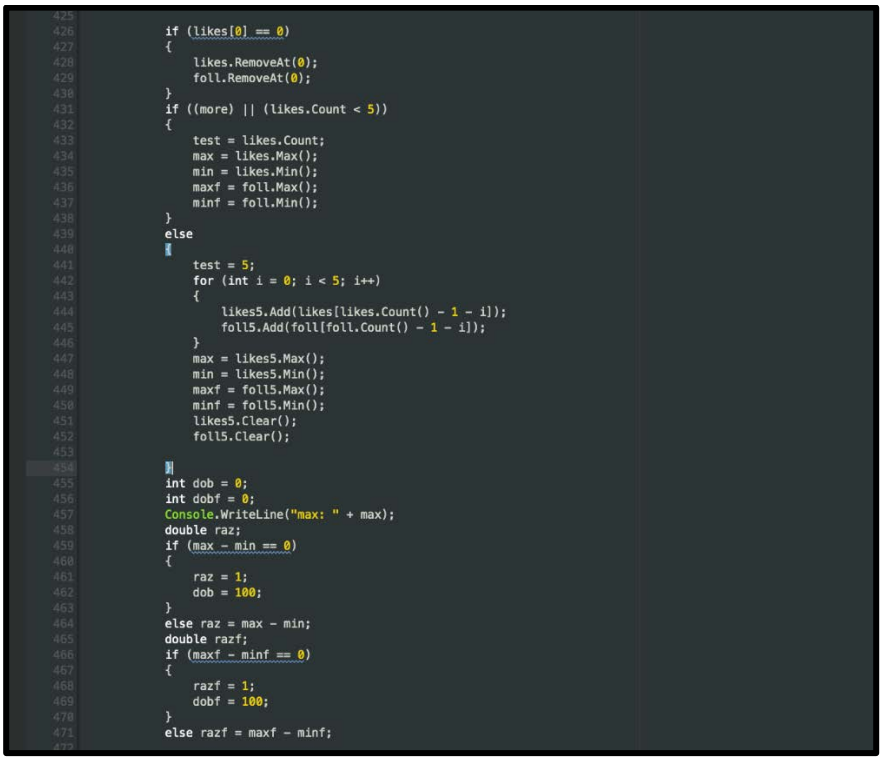
Рисунок 50 — Метод Stats. Часть 1



```
if (first && File.Exists("users/"  + Username + "2.txt"))
{
    foreach (string s in File.ReadLines("users/" + Username + "2.txt"))
    {
        if (s != "")
        {
            likes.Add(Convert.ToInt32(s.Split('%')[0]));
            dates.Add(s.Split('%')[1]);
            foll.Add(Convert.ToInt32(s.Split('%')[2]));
        }
        Console.WriteLine(s);
    }
    Console.WriteLine("12321");

}
Console.WriteLine(Username);
var feed = _instaApi.GetUserMediaAsync(Username, PaginationParameters.Empty).Result.Value;
Thread.Sleep(500);
Console.WriteLine(Username);
int lk = 0;
int folk = 0;
foreach (var media in feed)
{
    lk = lk + media.LikesCount;
}

Console.WriteLine(Username);
folk = _instaApi.GetUserFollowersAsync(Username, PaginationParameters.Empty).Result.Value.Count();
if (lk != likes[likes.Count() - 1] || folk != foll[foll.Count() - 1])
{
    likes.Add(lk);
    foll.Add(folk);
    dates.Add(DateTime.Today.ToShortDateString());
    File.AppendAllText("users/"  + Username + "2.txt", lk + "%" + DateTime.Today.ToShortDateString() + "%" + folk + "\n");

}
Console.WriteLine(folk);
Thread.Sleep(500);
double max = 0;
double min = 0;
double maxf = 0;
double minf = 0;
int test = 0;
```

Рисунок 51 — Метод Stats. Часть 2

После составления списков начинается процесс максимальных и минимальных значений, вычисляются отступы. Все это необходимо для корректного составления графиков (рисунок 52).



```
if (likes[0] == 0)
{
    likes.RemoveAt(0);
    foll.RemoveAt(0);
}
if ((more) || (likes.Count < 5))
{
    test = likes.Count;
    max = likes.Max();
    min = likes.Min();
    maxf = foll.Max();
    minf = foll.Min();
}
else
{
    test = 5;
    for (int i = 0; i < 5; i++)
    {
        likes5.Add(likes[likes.Count() - 1 - i]);
        foll5.Add(foll[foll.Count() - 1 - i]);
    }
    max = likes5.Max();
    min = likes5.Min();
    maxf = foll5.Max();
    minf = foll5.Min();
    likes5.Clear();
    foll5.Clear();

}
int dob = 0;
int dobf = 0;
Console.WriteLine("max: " + max);
double raz;
if (max - min == 0)
{
    raz = 1;
    dob = 100;
}
else raz = max - min;
double razf;
if (maxf - minf == 0)
{
    razf = 1;
    dobf = 100;
}
else razf = maxf - minf;
```

Рисунок 52 — Метод Stats. Часть 3

После этого, по специально созданной формуле, вычисляются координаты расположения для всех данных из списка. На оси абсцисс данные располагаются по дате запроса, на оси ординат по значению данных. Так же если при отправке запроса новые данные не отличаются от последних, то эти данные не учитываются.

Создаются кисти: зеленая для количества оценок, синяя для количества подписчиков (рисунок 53).

```
474             for (int i = 0; i < test; i++)
475             {
476                 // pix.Add(Convert.ToInt32(likes[likes.Count - 1 - i] / max * 100 * 2));
477                 pix.Add(Convert.ToInt32((likes[likes.Count - 1 - i] - min) * 200 / raz) + dob);
478                 follk.Add(Convert.ToInt32((foll[foll.Count - 1 - i] - minf) * 200 / razf) + dobf);
479                 dat.Add(dates[dates.Count - 1 - i]);
480                 lik.Add(likes[likes.Count - 1 - i]);
481                 fo11.Add(foll[foll.Count - 1 - i]);
482 
483             }
484             Thread.Sleep(500);
485 
486             Pen gp = new Pen(Brushes.Green)
487             {
488                 Width = 2.0F
489             };
490             Pen gp2 = new Pen(Brushes.Blue)
491             {
492                 Width = 2.0F
493             };
494 
```

Рисунок 53 — Метод Stats. Часть 4

Затем начинает создаваться рисунок. Сначала рисуются оси, задаются шрифты и наносятся надписи. Также на этом моменте задаются изначальные координаты и отступы (рисунок 54).

```
497             using (var gr = Graphics.FromImage(bmp))
498             {
499                 gr.FillRectangle(Brushes.White, new Rectangle(0, 0, bmp.Width, bmp.Height));
500                 gr.FillRectangle(Brushes.Black, new Rectangle(10, bmp.Height - 20, bmp.Width - 25, 5));
501                 gr.FillRectangle(Brushes.Black, new Rectangle(10, bmp.Height - 370, bmp.Width - 25, 5));
502                 gr.FillRectangle(Brushes.Black, new Rectangle(10, 10, 5, bmp.Height - 25));
503                 int c = 0;
504                 int k = 30;
505                 int x = 20, y = bmp.Height - 20, x1 = 0, y1 = 0;
506                 pix.Reverse();
507                 dat.Reverse();
508                 lik.Reverse();
509                 fo11.Reverse();
510 
511 
512 
513                 SolidBrush drawBrush = new SolidBrush(Color.Black);
514                 PointF strname = new PointF(30, 365);
515                 Font drawFont3 = new Font("Arial", 25);
516                 gr.DrawString("Общее количество лайков на профиле", drawFont3, drawBrush, strname);
517 
518                 PointF strname2 = new PointF(30, 15);
519                 Font drawFont4 = new Font("Arial", 25);
520                 gr.DrawString("Количество подписчиков на профиле", drawFont4, drawBrush, strname2);
521 
```

Рисунок 54— Метод Stats. Часть 5

На график наносятся значения количества оценок за все запросы. Они располагаются по заранее просчитанным координатам, которые зависят от количества данных и разницы между минимальным и максимальным значением (рисунок 55).

```
522
523             foreach (int p in pix)
524             {
525                 x1 = k + 5;
526                 y1 = bmp.Height - p - 30 + 5;
527                 gr.DrawLine(gp, x, y, x1, y1);
528                 gr.FillEllipse(Brushes.Green, new Rectangle(k, bmp.Height - p - 30, 10, 10));
529                 if (more)
530                 {
531                     if ((c == procentplus && c < pix.Count - 3) || c == pix.Count - 1)
532                     {
533                         PointF po = new PointF(k, bmp.Height - p - 50);
534                         PointF pod = new PointF(k, bmp.Height - 12);
535
536
537
538                         Font drawFont = new Font("Arial", 11);
539                         Font drawFont2 = new Font("Arial", 8);
540
541
542                         double t;
543                         t = lik[c];
544                         gr.DrawString("" + t, drawFont, drawBrush, po);
545                         gr.DrawString("" + dat[c], drawFont2, drawBrush, pod);
546
547                         procentplus = procentplus + 5;
548                     }
549
550                 }
551                 else
552                 {
553
554                     PointF po = new PointF(k, bmp.Height - p - 50);
555                     PointF pod = new PointF(k, bmp.Height - 12);
556
557
558
559                     Font drawFont = new Font("Arial", 11);
560                     Font drawFont2 = new Font("Arial", 8);
561
562
563                     double t;
564                     t = lik[c];
565                     gr.DrawString("" + t, drawFont, drawBrush, po);
566                     gr.DrawString("" + dat[c], drawFont2, drawBrush, pod);
567                 }
568                 x = k;
569                 y = bmp.Height - p - 30 + 5;
570
571                 if (more) k = k + 600 / (pix.Count + 3);
572                 else k = k + 600 / 5;
573                 c++;
574                 Thread.Sleep(500);
575
576
577             }
```

Рисунок 55 — Метод Stats. Часть 6

После нанесения графика оценок, координаты обновляются на новые начальные координаты для графика подписчиков и происходит такая же процедура. Если пользователь отправляет запрос на статистику за все свои запросы и если таких запросов много, то бот автоматически убирает некоторые значения во избежание нагромождения данных друг на друга (рисунок 56). Если со времени последнего запроса данные не изменились, то они не будут учитываться при составлении графика.

```
            c = 0;
            k = 30;
            procentplus = 0;
            x = 20; y = bmp.Height / 2 - 20; x1 = 0; y1 = 0;
            follk.Reverse();
            foreach (int f in follk)
            {
                x1 = k + 5;
                y1 = bmp.Height / 2 - f - 30 + 5;
                gr.DrawLine(gp2, x, y, x1, y1);
                gr.FillEllipse(Brushes.Blue, new Rectangle(k, bmp.Height / 2 - f - 30, 10, 10));

                if (more)
                {
                    if ((c == procentplus && c < follk.Count - 3) || c == follk.Count - 1)
                    {
                        PointF po = new PointF(k, bmp.Height / 2 - f - 50);
                        PointF pod = new PointF(k, bmp.Height / 2 - 12);



                        Font drawFont = new Font("Arial", 11);
                        Font drawFont2 = new Font("Arial", 8);

                        double t;
                        t = fo11[c];
                        gr.DrawString("" + t, drawFont, drawBrush, po);
                        gr.DrawString("" + dat[c], drawFont2, drawBrush, pod);
                        procentplus = procentplus + 5;

                    }
                }
                else
                {
                    PointF po = new PointF(k, bmp.Height / 2 - f - 50);
                    PointF pod = new PointF(k, bmp.Height / 2 - 12);



                    Font drawFont = new Font("Arial", 11);
                    Font drawFont2 = new Font("Arial", 8);


                    double t;
                    t = fo11[c];
                    gr.DrawString("" + t, drawFont, drawBrush, po);
                    gr.DrawString("" + dat[c], drawFont2, drawBrush, pod);
                }
                x = k;
                y = bmp.Height / 2 - f - 30 + 5;


                if (more) k = k + 600 / (follk.Count + 3);
                else k = k + 600 / 5;

                c++;
                Thread.Sleep(500);


            }
```

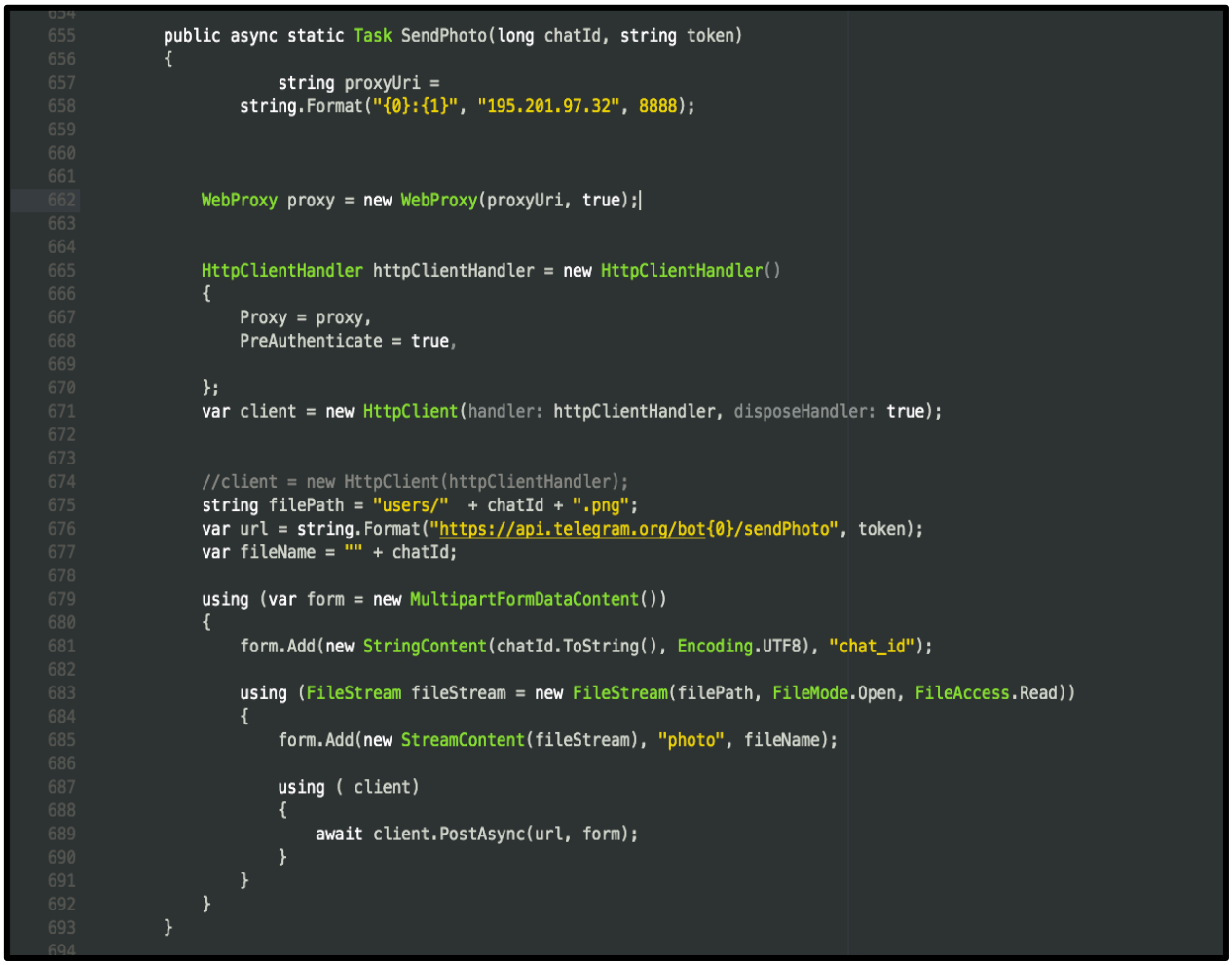
Рисунок 56 — Метод Stats. Часть 7

После создания полноценного графика со всеми необходимыми данными изображение сохраняется в специальную папку пользователя. Для идентификации изображения используется его id. Вызывается вспомогательный метод SendPhoto и вычисляется среднее количество оценок на фото, которое будет отправлено отдельным сообщением (рисунок 57).

```
            var path = "users/"  + chatid + ".png";
            bmp.Save(path);
            pix.Reverse();
            dat.Reverse();
            fo11.Reverse();
            follk.Reverse();
            SendPhoto(ChatId, TelegramToken).Wait();


        }
        prog.SendMessage(ChatId, "Среднее количество лайков на фото: " + Convert.ToInt32(lk / feed.Count()));
        Username = _instaApi.GetCurrentUserAsync().Result.Value.UserName;
        anoth = false;
        another = null;

    }
```

Рисунок 57 — Метод Stats. Часть 8

SendPhoto (рисунок 58) выполняет функцию отправки созданного ранее графика статистики профиля пользователю.



```
    public async static Task SendPhoto(long chatId, string token)
    {
                string proxyUri =
            string.Format("{0}:{1}", "195.201.97.32", 8888);



        WebProxy proxy = new WebProxy(proxyUri, true);


        HttpClientHandler httpClientHandler = new HttpClientHandler()
        {
            Proxy = proxy,
            PreAuthenticate = true,

        };
        var client = new HttpClient(handler: httpClientHandler, disposeHandler: true);


        //client = new HttpClient(httpClientHandler);
        string filePath = "users/"  + chatId + ".png";
        var url = string.Format("https://api.telegram.org/bot{0}/sendPhoto", token);
        var fileName = "" + chatId;

        using (var form = new MultipartFormDataContent())
        {
            form.Add(new StringContent(chatId.ToString(), Encoding.UTF8), "chat_id");

            using (FileStream fileStream = new FileStream(filePath, FileMode.Open, FileAccess.Read))
            {
                form.Add(new StreamContent(fileStream), "photo", fileName);

                using ( client)
                {
                    await client.PostAsync(url, form);
                }
            }
        }
    }
```

Рисунок 58 — Метод SendPhoto

Метод Info компонует все данные пользователя в одно сообщение и отправляет его в простой для понимания форме (рисунок 59).

```
public void Info()
{
    string t = "", b = "", mi = "", ma = "", sp = "", ti = "";

    if (x1000)
    {
        ti = "Привышен";
    }
    else ti = "Не привышен";


    if (BlackList.Count == 0) { b = "Не задан"; }
    else
    {
        foreach (string bs in BlackList)
        {
            b = b + bs + ",";
        }
        b = b.Trim(',');
    }
    if (Tag.Count == 0) { t = "Не задан"; }
    else
    {
        foreach (string bs in Tag)
        {
            t = t + bs + ",";
        }
        t = t.Trim(',');
    }
    if (MinLike == 0) mi = "Не задан";
    else mi = MinLike + "";
    if (MaxLike == 10000000) ma = "Не задан";
    else ma = MaxLike + "";

    if (SendPics) sp = "Да";
    else sp = "Нет";
    prog.SendMessage(ChatId, "Логин : " + Username + "\nМин лайков : " + mi + "\nМакс лайков : " + ma + "\nЛайков по тегу : " + TagCount + "\nТеги : " + t + "\nBlacklist : " + b + "\nПоказывать фото : " + sp +

}
```

Рисунок 59 — Метод Info

Clear, ClearBL и ClearTag отвечают за очистку статистики, черного списка и списка тегов соответственно (рисунок 60).

```
        public void Clear()
        {

            if (File.Exists(Username + "2.txt"))
            {

                File.Delete(Username + "2.txt");
                likes.Clear();
                dates.Clear();
                foll.Clear();
                first = true;


            }
        }
        public void ClearBL()
        {
            BlackList.Clear();
            prog.SendMessage(ChatId, "Blacklist очищен");
        }

        public void ClearTag()
        {
            Tag.Clear();
            prog.SendMessage(ChatId, "Тэги очищены");
        }
```

Рисунок 60 — Методы Clear, ClearBL и ClearTag

Последний метод в User.cs — Logout, он необходим для деавторизации пользователя, если тот хочет сменить аккаунт (рисунок 61).

```
764         public void Logout()
765         {
766             if (Username != null)
767             {
768                 _instaApi.LogoutAsync();
769                 prog.SendMessage(ChatId, "Вы вышли из профиля");
770                 isauto = false;
771             }
772             else prog.SendMessage(ChatId, "Вы еще не авторизировались");
773         }
774
```

Рисунок 61 — Метод Logout

2.3.4 Оформление бота

В качестве элементов внешнего оформления бота выступают:

- описание, демонстрируемое пользователю при первом знакомстве с ботом (рисунок 62);
- первое сообщение, которое пользователь получает автоматически при начале работы с ботом (рисунок 63);
- изображение, использующееся для упрощения поиска бота среди переписок (рисунок 64);
- имя бота.

В описании указаны основные возможности, название и принцип работы бота.

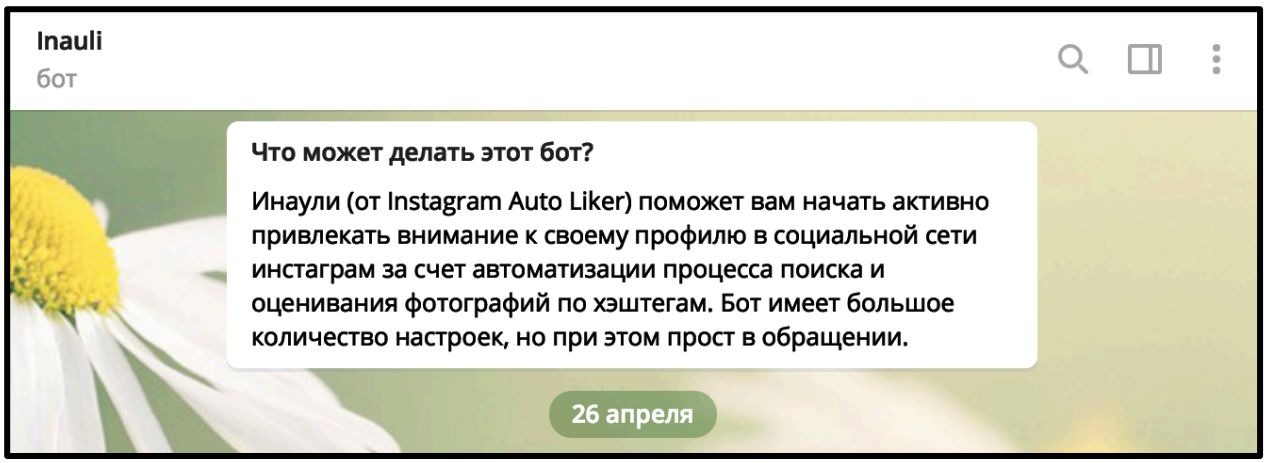


Рисунок 62 — Описание бота

Первое сообщение содержит в себе приветствие и в неформальной манере объясняет, как пользоваться ботом.

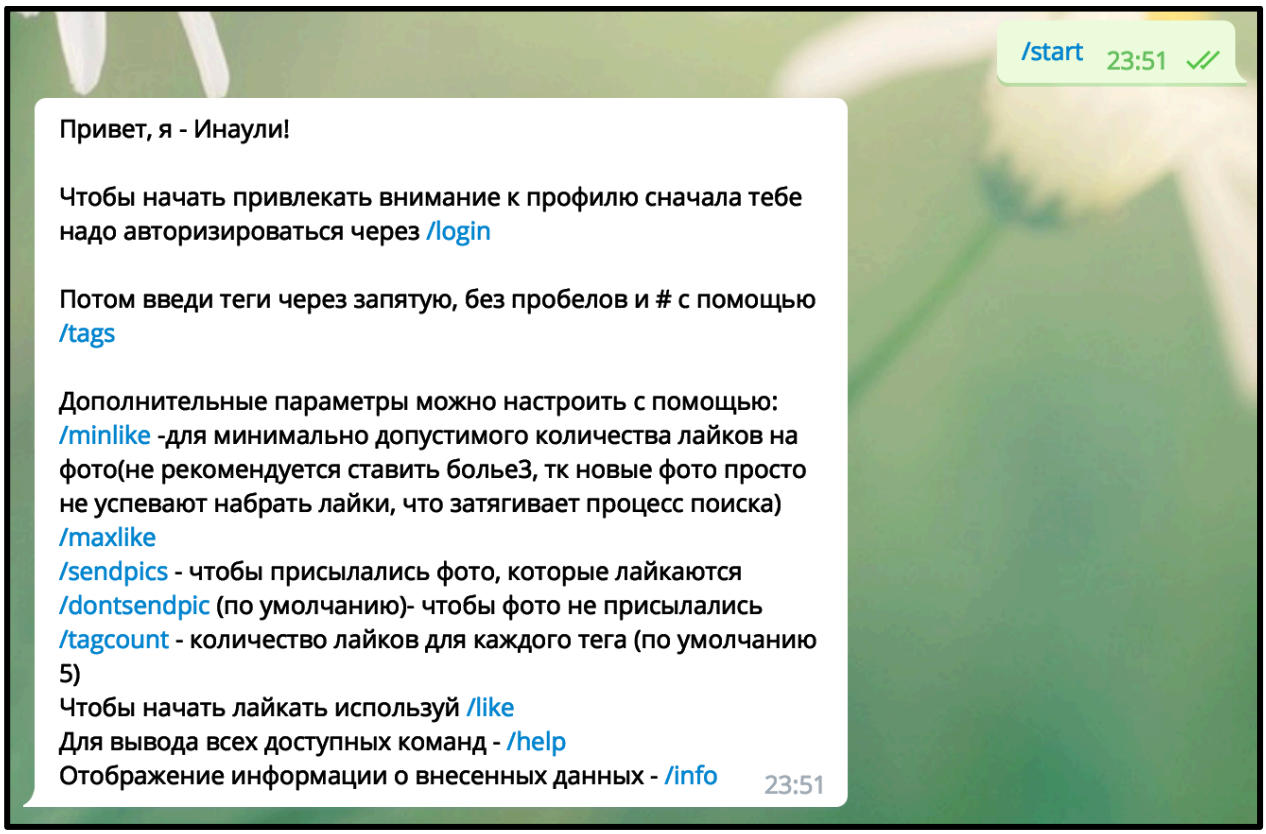


Рисунок 63 — Первое сообщение

Для отличительного изображения было принято решение объединить логотипы Instagram и Telegram, так бот служит связующим звеном между этими сервисами и это напрямую отражает его суть.

Рисунок 64 — Логотипы Instagram, Telegram и Inauli

В качестве имени была выбрана аббревиатура от Instagram Auto Liker — Inauli. Такое имя очеловечивает бот в глазах пользователя и создает ощущение прямого общения.

2.4 Апробирование Telegram-бота

Для тестирования работоспособности и полезности бота был создан тестовый аккаунт в Instagram (рисунок 65). Чтобы обеспечить максимальную

чистоту эксперимента к этому аккаунту не применялись никакие иные методы развития, кроме созданного бота. Аккаунт посвящен коту, соответственно поиск и обработка фотографий шли по тегам: cat, cats, instacat, catstagram, pet, pets. По каждому тегу оценивалось по 100–150 фотографий 1–2 раза в неделю в течение месяца.

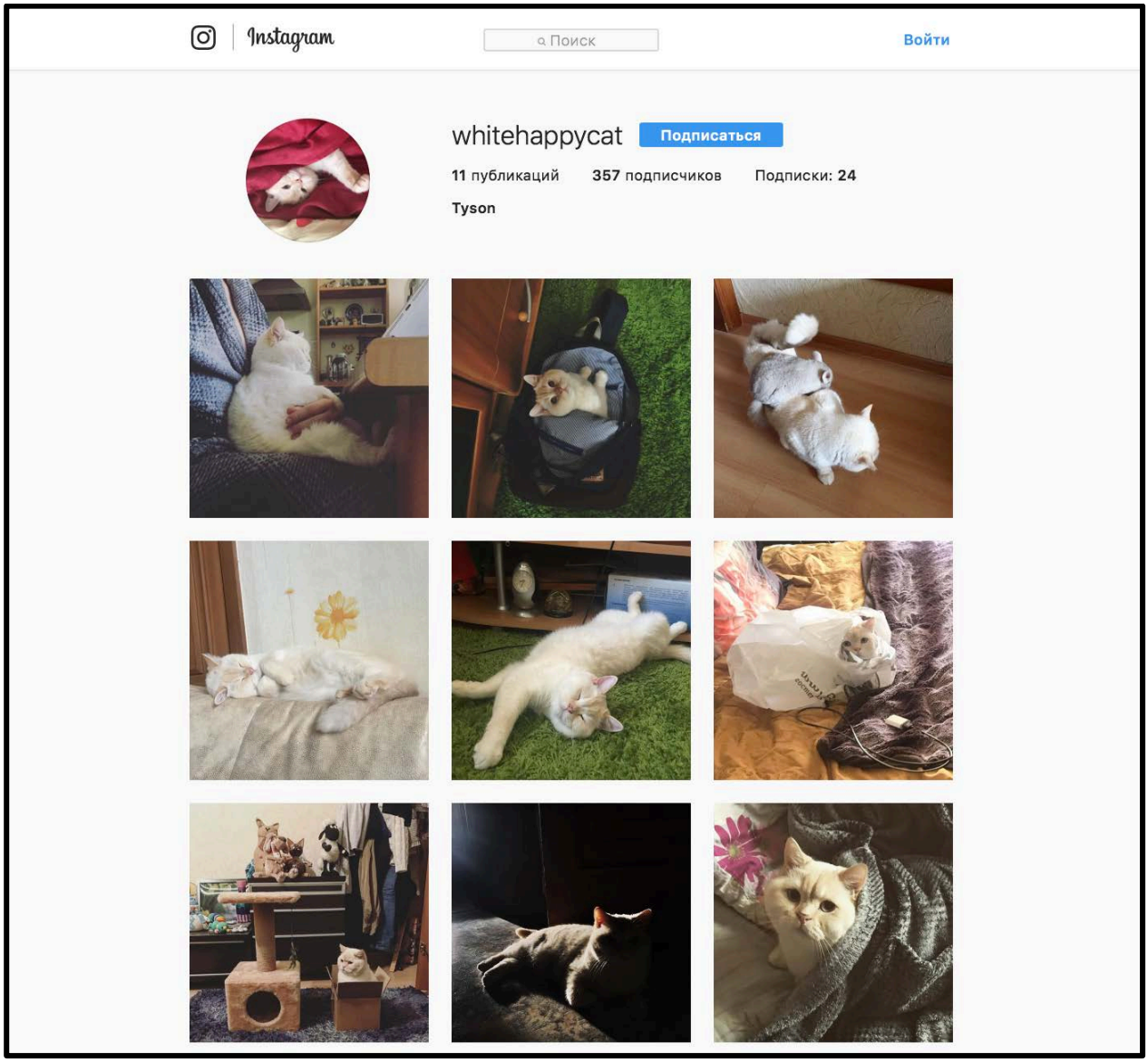


Рисунок 65 — Профиль whiTehappycat

В результате новый, практически неактивный профиль смог собрать почти 400 подписчиков, среднее количество оценок под фотографией возросло до 251 (рисунок 66). За время проведения эксперимента приложение Instagram открывалось исключительно для выкладывания новых фотографий

и ответов на комментарии пользователей, вся остальная деятельность велась с помощью бота.

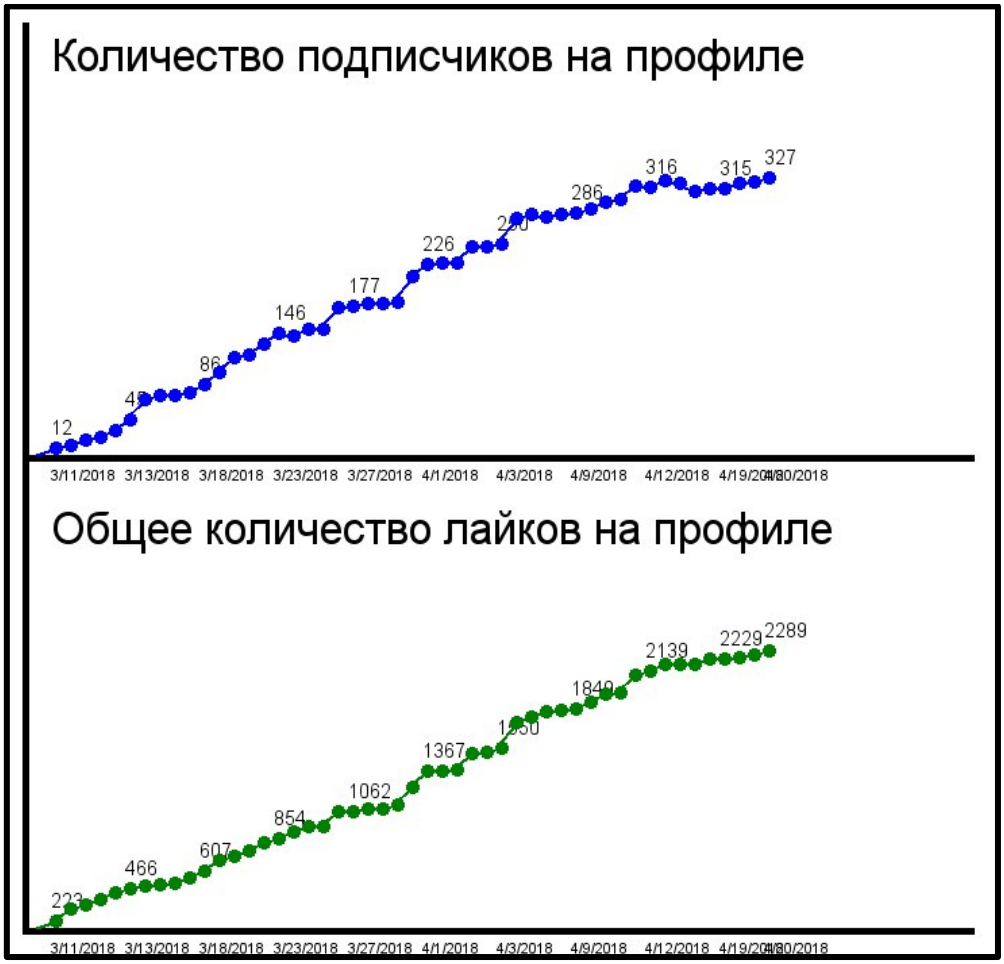


Рисунок 66 — Статистика профиля whiTehappycat

Совмещая бота с другими методами развития аккаунта можно достичь подобного результата за гораздо меньшее время. Но так как целью эксперимента было проверить полезность проекта, другие методы не применялись.

После удачного эксперимента было принято решение опробовать бота не на личном аккаунте. В использовании Telegram-бота был заинтересован питомник по продаже котов породы мейн кун Dias MonTes. На момент начала работы аккаунт имел 118 подписчиков, среднее количество оценок под фото составляло 17 лайков. Менее чем за неделю количество подписчиков было увеличено на 45.76 % от начального и среднее количество оценок поднялось почти в 2 раза. Общее количество оценок на профиле возросло с 498 до 813 (рисунок 67).

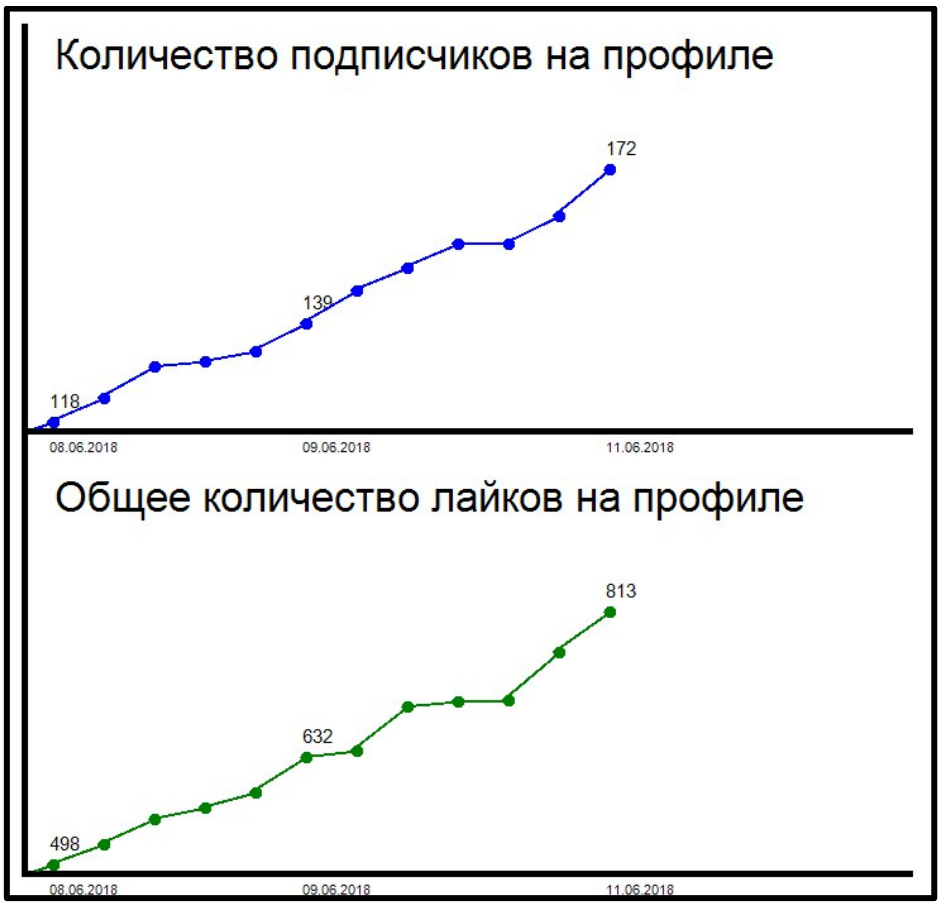


Рисунок 67 — Статистика профиля diasmainecoon

Этот эксперимент также показал отличные результаты. Заказчик, в лице Е. Н. Журавлевой, остался доволен (рисунок 68). На данном этапе работа над продвижением аккаунта питомника не будет завершена и Inauli продолжит развивать этот профиль в Instagram.

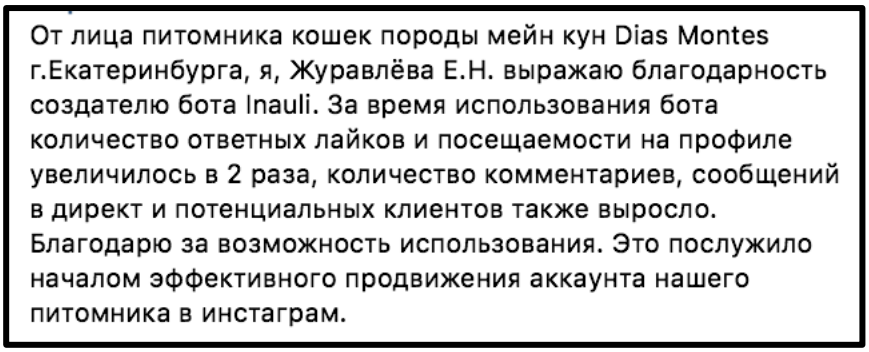

От лица питомника кошек породы мейн кун Dias Montes г.Екатеринбурга, я, Журавлёва Е.Н. выражаю благодарность создателю бота Inauli. За время использования бота количество ответных лайков и посещаемости на профиле увеличилось в 2 раза, количество комментариев, сообщений в директ и потенциальных клиентов также выросло. Благодарю за возможность использования. Это послужило началом эффективного продвижения аккаунта нашего питомника в инстаграм.

Рисунок 68 — Отзыв о работе Inauli

ЗАКЛЮЧЕНИЕ

Интернет уже давно стал частью повседневной жизни людей, и теперь, когда кто-то открывает свое дело или начинает работать над новым проектом, ему просто необходимо продвигать свое дело в социальных сетях. Для больших компаний, вроде Coca Cola или Uber, не составляет труда тратить десятки или даже сотки тысяч долларов на рекламу, но кроме таких гигантов в мире очень много простых людей без стартового капитала. Таким людям, в начале их деятельности в Интернете, необходимо решение, которое позволит привлечь внимание других пользователей к тому, что они делают. При этом такое решение не должно приводить к трудностям в финансовом плане. Поэтому Telegram-бот, который будет работать на любых платформах, без каких-либо вложений является актуальной темой.

Было написано около 1411 строк кода.

В ходе работы были выполнены следующие задачи:

- проанализирована предметная область;
- выявлены плюсы и минусы существующих решений;
- изучены средства для реализации;
- разработан алгоритм реализации проекта;
- сформулированы требования к продукту;
- создан Telegram-бот в программе Visual Studio;
- создан логотип для Telegram-бота;
- зарегистрирован и полностью настроен бот Inauli в программе Telegram;
- Telegram-бот протестирован на тестовом и обычном аккаунтах в сети Instagram.

Таким образом, задачи выпускной квалификационной работы выполнены, цель работы достигнута.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Бот (программа) [Электронный ресурс]. — Режим доступа: https://ru.wikipedia.org/wiki/Бот_(программа)#Чат-боты (дата обращения: 27.11.2017).

2. Документация Telegram: Примеры ботов [Электронный ресурс]. — Режим доступа: https://tlgrm.ru/docs/bots/samples#c-sharp (дата обращения: 11.03.2018).

3. Использование прокси на C# [Электронный ресурс]. — Режим доступа: http://lsreg.ru/ispolzovanie-proksi-na-c/ (дата обращения: 13.03.2018).

4. Использование таймеров [Электронный ресурс]. — Режим доступа: https://metanit.com/sharp/tutorial/11.9.php (дата обращения: 23.04.2018).

5. Как зашифровать и расшифровать файл с помощью C# [Электронный ресурс]. — Режим доступа: https://social.msdn.microsoft.com/Forums/ru-RU/2324247d-93d8-489f-acbe-8fb18463f240/105010721082-1079107210961080109210881086107410721090110 0?forum=programminglanguageru (дата обращения: 05.04.2018).

6. Как избежать бана и блокировки в Instagram [Электронный ресурс]. — Режим доступа: https://instaplus.freshdesk.com/support/solutions/articles/6000153057-Урок-3-Как-избежать-бана-и-блокировки-в-instagram-Лимиты-и-рекомендации- (дата обращения: 23.05.2018).

7. Класс Graphics (SysTem.Drawing) [Электронный ресурс]. — Режим доступа: https://msdn.microsoft.com/ru-ru/library/sysTem.drawing.graphics(v=vs.110).aspx (дата обращения: 15.04.2018).

8. Лимиты и ограничения для Instagram [Электронный ресурс]. — Режим доступа: http://smorovoz.ru/instagram/limit-ogranichenie-instagram.html (дата обращения: 17.03.2018).

9. Отправляем сообщения в Telegram из C# [Электронный ресурс]. — Режим доступа: https://msdn.microsoft.com/ru-ru/mt650910.aspx (дата обращения: 15.02.2018).

10. Пространство имен SysTem.Security.Cryptography [Электронный ресурс]. — Режим доступа: https://msdn.microsoft.com/ru-ru/library/sysTem.security.cryptography(v=vs.110).aspx (дата обращения: 13.03.2018).

11. Работа с потоками в C# [Электронный ресурс]. — Режим доступа: https://rsdn.org/article/dotnet/CSThreading1.xml (дата обращения: 13.03.2018).

12. Сериализация [Электронный ресурс]. — Режим доступа: https://metanit.com/sharp/tutorial/6.1.php (дата обращения: 19.04.2018).

13. Чтение и запись файла. Класс FileStream [Электронный ресурс]. — Режим доступа: https://metanit.com/sharp/tutorial/5.4.php (дата обращения: 03.04.2018).

14. Эффективность наружной рекламы и перспективы ее развития [Электронный ресурс]. — Режим доступа: https://www.marketing.spb.ru/lib-special/case/outdoor_adv.htm (дата обращения: 12.05.2018).

15. Bot Code Examples — Telegram APIs [Электронный ресурс]. — Режим доступа: https://core.Telegram.org/bots/samples (дата обращения: 21.03.2018).

16. Какая реклама самая эффективная [Электронный ресурс]. — Режим доступа: https://www.leader-web.ru/blog/marketing/sravnenie_reklami/ (дата обращения: 02.03.2018).

17. C# Инструментальное ПО [Электронный ресурс]. — Режим доступа: http://www.intuit.ru/studies/courses/3632/874/lecture/14295?page=3 (дата обращения: 18.04.2018).

18. C# для вундеркиндов [Электронный ресурс]. — Режим доступа: https://msdn.microsoft.com/ru-ru/library/bb330922(v=vs.80).aspx (дата обращения: 11.02.2018).

19. Fireseo [Электронный ресурс]. — Режим доступа: https://fireseo.ru (дата обращения: 21.05.2018).

20. How To Create Telegram Bot — C# [Электронный ресурс]. — Режим доступа: https://www.youtube.com/watch?v=RWMiYIGOjRI (дата обращения: 05.03.2018).

21. Instagram [Электронный ресурс]. — Режим доступа: https://ru.wikipedia.org/wiki/Instagram (дата обращения: 18.03.2018).

22. Instaplus [Электронный ресурс]. — Режим доступа: https://instaplus.me (дата обращения: 21.05.2018).

23. InstaSharper: Private Instagram API [Электронный ресурс]. — Режим доступа: https://github.com/a-legotin/InstaSharper (дата обращения: 13.03.2018).

24. JetInsta [Электронный ресурс]. — Режим доступа: https://jet-insta.com/ru/ (дата обращения: 21.05.2018).

25. Quickstart: Install and use a package in Visual Studio [Электронный ресурс]. — Режим https://docs.microsoft.com/en-us/nuget/quickstart/install-and-use-a-package-in-visual-studio (дата обращения: 13.03.2018).

26. SocialHammer [Электронный ресурс]. — Режим доступа: https://socialhammer.com (дата обращения: 21.05.2018).

27. System.Drawing Пространство имен [Электронный ресурс]. — Режим доступа: https://msdn.microsoft.com/ru-ru/library/sysTem.drawing (v=vs.110).aspx (дата обращения: 15.04.2018).

28. System.Runtime.Serialization.Formatters.Binary Пространство имен [Электронный ресурс]. — Режим доступа: https://msdn.microsoft.com/ru-ru/library/sysTem.runtime.serialization.formatTers.binary(v=vs.110).aspx (дата обращения: 19.04.2018).

29. Telegram [Электронный ресурс]. — Режим доступа: https://ru.wikipedia.org/wiki/Telegram (дата обращения: 03.03.2018).

30. Web-прокси [Электронный ресурс]. — Режим доступа: https://professorweb.ru/my/csharp/web/level2/2_6.php (дата обращения: 18.05.2018).

ПРИЛОЖЕНИЕ

Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

«Российский государственный профессионально-педагогический университет»

Институт инженерно-педагогического образования

Кафедра информационных систем и технологий

Направление подготовки 09.03.02 Информационные системы и технологии

Профиль подготовки «Информационные технологии в медиаиндустрии»

УТВЕРЖДАЮ
Заведующий кафедрой

_____ Н.С. Толстова
подпись и.о. фамилия

« ___ » _____ 2018 г.

ЗАДАНИЕ
на выполнение выпускной квалификационной работы бакалавра

студента (ки) *4* курса группы *ИТм-402*
Шулепова Дмитрия Николаевича
фамилия, имя, отчество полностью

1. Тема Telegram-бот для продвижения бизнес-аккаунтов в социальной сети Instagram
утверждена распоряжением по институту от «_____» _____ 20 г. № ___
2. Руководитель Толстова Наталья Сергеевна
фамилия, имя, отчество полностью

доцент (ученая степень) *к.пед.н.* (ученое звание) *доцент кафедры ИС* (должность) *РГППУ* (место работы)

3. Место преддипломной практики РГППУ

4. Исходные данные к ВКР _____

5. Содержание текстовой части ВКР (перечень подлежащих разработке вопросов)

6. Перечень демонстрационных материалов презентация выполненная в MS Power Point, обзорный видеоролик по работе telegram-бота, telegram-бот для продвижения бизнес-аккаунтов в социальной сети Instagram.

7. Календарный план выполнения выпускной квалификационной работы

№ п/п	Наименование этапа дипломной работы	Срок выполнения этапа	Процент выполнения ВКР	Отметка руководителя о выполнении
1	Сбор информации по выпускной квалификационной работе	23.04.2018	10%	подпись
2	Выполнение работ по разрабатываемым вопросам и их изложение в пояснительной записке:	03.05.2018	60%	подпись
2.1	Анализ существующих разработок по теме.	03.05.2018	5%	подпись
2.2	Написание основного кода telegram-бота	04.05.2018	30%	подпись
2.3	Тестирование и дополнение функционала	07.05.2018	5%	подпись
2.4	Апробирование telegram-бота	11.05.2018	10%	подпись
2.5	Сбор статистики	14.05.2018	10%	подпись
3	Оформление текстовой части ВКР	15.05.2018	10%	подпись
4	Выполнение демонстрационных материалов к ВКР	01.06.2018	10%	подпись
5	Нормоконтроль	06.06.2018	5%	подпись
6	Подготовка доклада к защите в ГЭК	13.06.2018	5%	подпись

8. Консультанты по разделам выпускной квалификационной работы

Наименование раздела	Консультант	Задание выдал		Задание принял	
		подпись	дата	подпись	дата

Руководитель _____ _____ *Задание получил* _____ _____
подпись дата подпись студента дата

9. Дипломная работа и все материалы проанализированы.
Считаю возможным допустить Шулепова Д.Н. к защите выпускной квалификационной работы в государственной экзаменационной комиссии.
Руководитель _____ _____
подпись дата

10. Допустить Шулепова Д.Н. к защите выпускной квалификационной работы
фамилия и. о. студента
в государственной экзаменационной комиссии (протокол заседания кафедры от «___» _____ 20___г., № _____)
Заведующий кафедрой _____ _____
подпись дата