

Программное обеспечение для автоматизированной покупки товаров лимитированной серии

Авторы:

Елисеев Владимир Андреевич,
11 кл. МБОУ ЛСТУ №2 г. Пензы

Забенько Вячеслав Борисович,
11 кл. МБОУ ЛСТУ №2 г. Пензы

Киреев Никита Андреевич,
11 кл. МБОУ ЛСТУ №2 г. Пензы

Петров Егор Дмитриевич,
11 кл. МБОУ ЛСТУ №2 г. Пензы

Научное руководство:

Адамский С. С., учитель
информатики высшей категории
МБОУ ЛСТУ №2 г. Пензы

Голикова И. А., учитель
информатики первой категории
МБОУ ЛСТУ №2 г. Пензы

Оглавление

Введение.....	3
1. Процедура покупки товара в «онлайне».....	4
2. Структура ПО и требования к его функциям.....	4
2.1 Структура ПО	4
2.2 «Авторегер»	5
2.3 «Монитор»	6
2.4 «Бот».....	8
Заключение	9

Введение

Информационные технологии ускоряют темп нашей жизни, делая обработку данных практически мгновенной. Современному человеку сегодня при помощи персональных компьютерных устройств совершенно нет необходимости посещать лично различные заведения социальной сферы, будь то административные государственные органы или же объекты торговли, в том числе и фирменные магазины.

В настоящий момент у крупных индустриальных организаций существует практика онлайн-распродаж своей продукции ограниченного выпуска, так называемой «лимитированной серии». Такие акции пользуются повышенным спросом на мировом уровне, поскольку сделки происходят дистанционно. Покупатели из разных городов и стран нашей планеты в установленные дату и время начала распродажи фирменных товаров буквально «обрушивают» сервера веб-сайтов продавцов с целью успешного приобретения товара. Поскольку товар ограничен в количестве, а спрос на фирменную продукцию избыточен – она распродается менее, чем за минуту (например, у крупных «брендов» по производству одежды и аксессуаров).

Для того, чтобы повысить шансы покупателя на успешную торговую сделку, существует способ автоматизации процесса покупки при помощи специального программного обеспечения (ПО) – «бота-скупщика», способного за доли секунды приобрести дефицитный товар. Именно такой подход и будет рассмотрен в настоящей работе.

Для достижения **главной цели работы** (создание автоматизированного ПО для упрощения и ускорения покупок в сети Интернет) были поставлены следующие **задачи**:

- 1) Проанализировать информацию по выбранной теме;
- 2) Разработать структуру ПО;
- 3) Выбрать средства разработки ПО;
- 4) Разработать и отладить ПО;
- 5) Апробировать ПО.

1. Процедура покупки товара в «онлайне»

Для реализации своей продукции крупные компании, как правило, используют свои онлайн-площадки – Интернет-магазины. Базовая структура таких площадок крайне проста: категории каталогов товаров, личные кабинеты пользователей, интегрированные системы онлайн-оплаты, «корзина».

На различных площадках используются разные системы оплаты. Одной из самых популярных систем является «Shopify», которая представлена преимущественно на европейских сайтах и сайтах в США. Есть и отечественные примеры, такие, как «Ю.касса» и ей подобные. Большинство платёжных комплексов используются для оплаты товаров различного типа, но есть и системы, которые созданы исключительно для конкретного типа товаров, например, «Mesh» и «FootSites» работают только на избранных площадках по продаже кроссовок и одежды.

Типичный алгоритм покупки товара в подобных системах выглядит следующим образом: пользователь посещает сайт площадки, выбирает товар, добавляет его в корзину, затем переходит на страницу заполнения личных данных (свой почтовый адрес, имя, и т.д.), выбирает тип оплаты, вводит данные своей карты и в завершении активирует процесс оплаты кнопкой «оплатить/оформить заказ». Если была выбрана оплата онлайн или опция «предварительный заказ», то дальше делается запрос на обработку платежа и платёжная система через банк подтверждает или отклоняет заказ.

Все вышеперечисленные действия можно упростить, если заранее заполнить свои данные в личном кабинете и сохранить при предварительной регистрации. Также есть специальные «расширения» для веб-браузеров, которые автоматически заполняют все пользовательские данные. Некоторые расширения даже сами нажимают кнопку «оплатить», как только она становится активной на веб-странице. В случае акции по лимитированной распродаже товара пользователь вынужден ожидать начала акции в определённое время и периодически выполнять запросы по обновлению статусов товара и/или акции.

2. Структура ПО и требования к его функциям

2.1 Структура ПО

Для автоматизации процесса покупки были выделены следующие составляющие для проекта:

- **«Авторегер»** – создаёт множество учётных записей для потенциального покупателя для увеличения шанса на успешную покупку;
- **«Монитор»** – отслеживает доступность желаемого товара на торговой площадке и оповещает пользователя;
- **«Бот»** – совершает покупку за пользователя с максимально возможной скоростью.

Исходя из специфики проблемы, разрабатываемое ПО должно обладать следующими программными возможностями:

- Симуляция действий пользователя на сайте;
- Парсинг веб-страниц;
- Отслеживание статусов товаров или акций;
- Авторегистрация учётных записей пользователей;
- Работа с сервисами по аренде виртуальных мобильных номеров через API;
- Автозапуск ПО в требуемое время.

2.2 «Авторегер»

Данная составляющая проекта реализована на языке JavaScript и скомпилировано в исполняемый файл при помощи конструктора скриптов «BAS» (Browser Automation Studio).

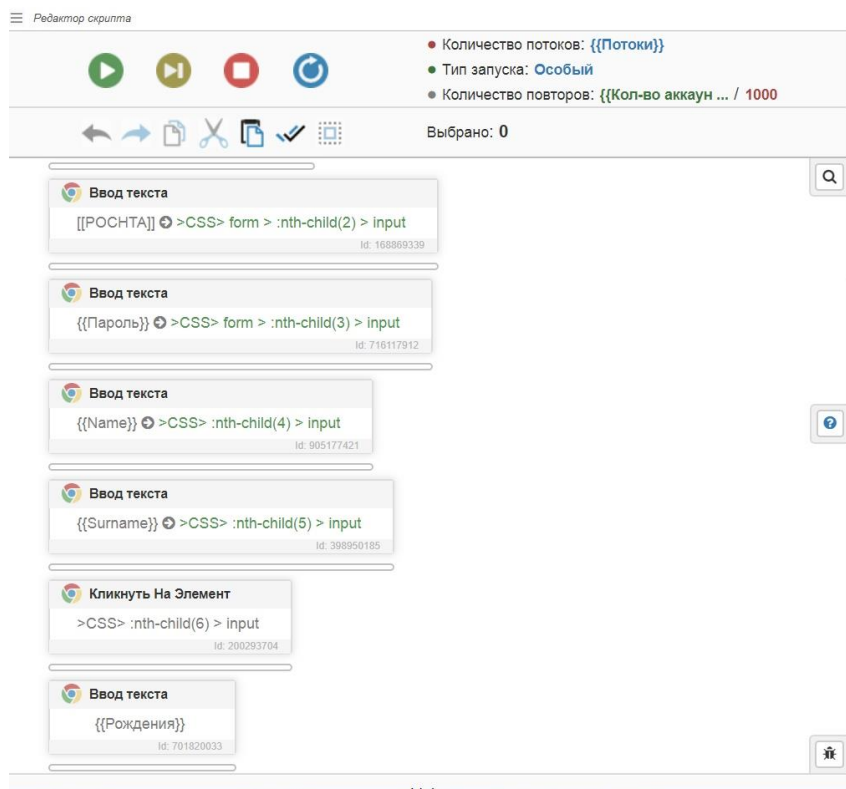


Рис. 1, фрагмент алгоритма в среде «BAS»

Для использования «авторегера» пользователю нужно обладать собственным доменным именем с подключенным сервисом почты для домена и настройкой переадресации писем с любых почтовых адресов такого доменного имени на один почтовый ящик. Это требование необходимо для того, чтобы у будущих автоматически-созданных учётных записей были уникальные адреса e-mail (но, внутри одного домена).

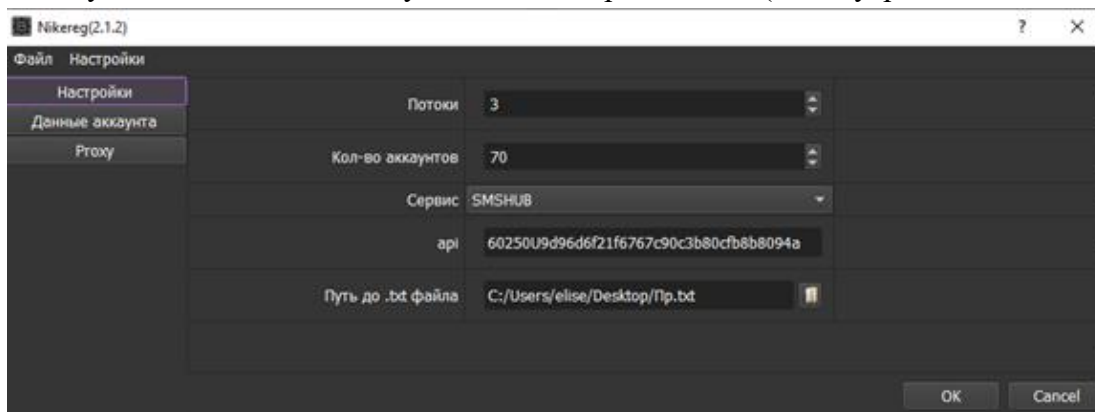


Рис. 2, фрагмент настройки «авторегера»

«Авторегер» получает на вход от пользователя его персональные данные, доменное имя для почты. При запуске процесса авторегистрации в фоновом многопоточном режиме создаются процессы веб-браузера, в каждом из которых скрипт выполняет заранее заложенные действия, направленных на создание новых учётных записей на торговой площадке.

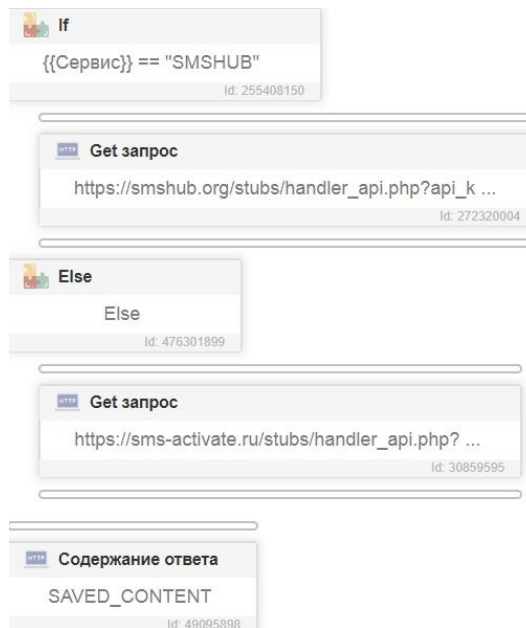


Рис. 3, фрагмент алгоритма «авторегера» в среде «BAS»

Так же в момент регистрации происходит аренда виртуального мобильного номера для подтверждения регистрации через входящее СМС-сообщение. Аренда происходит через взаимодействие ПО с API сервиса по аренде номеров: запрашивается наличие доступного номера и его идентификационные данные. После создания учётной записи на торговой площадке запрашивается содержимое входящего СМС-сообщения на виртуальный номер и происходит активация аккаунта для покупки товара. Принципиальный алгоритм работы «регера» представлен в приложении №1.

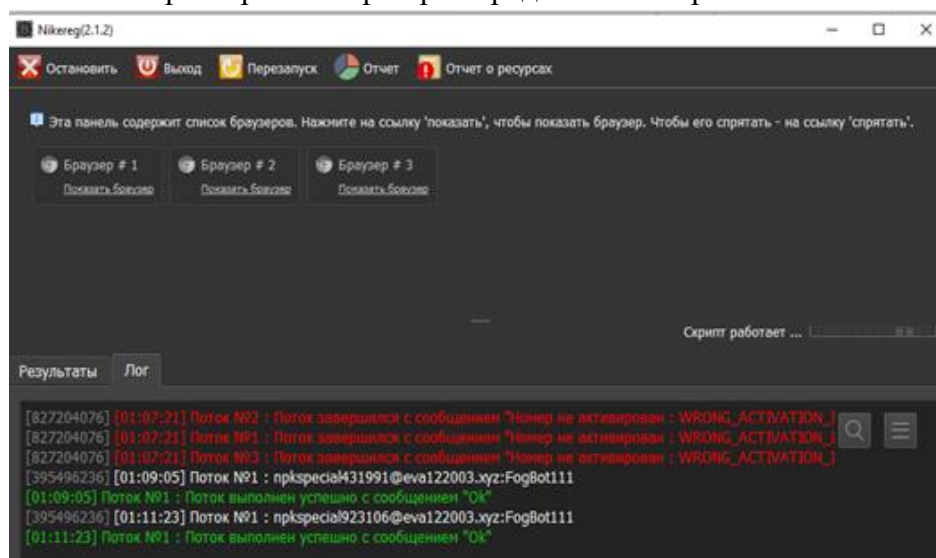


Рис. 4, фрагмент работы «авторегера»

В итоге работы «регера» создаётся текстовый файл с авторизационными данными от успешных регистраций аккаунта. Структура файла проста – текстовые данные, построчно записанные через символ-разделитель.

2.3 «Монитор»

Данная составляющая проекта реализуется при помощи языка python. Указанный язык программирования выбран из-за обширного количества доступных современных библиотек, документаций и обучающих примеров, что обуславливается популярностью Python в настоящее время.

При запуске «монитор» посылает запросы на сайт торговой площадки и получает в ответ код страницы в формате json, который легко обрабатывается одноимённой библиотекой.

```
html = requests.get(productslink, proxies = proxy_dict, headers=headers, timeout = 10)
products = json.loads(html.content)['products']
return products
```

Затем, в полученных данных выполняется выборка товаров (по ключевым словам), интересующих пользователя.

```
keywords = ['yeezy', 'dunk', 'air jordan 1', 'air force', 'force', 'aj 1', 'sacai']
proxy_num = thread_num * 4
while True:
    products = getproducts(getProxies(), getproductslink(shop_link), thread_num, proxy_num)
    try:
        for k in products:
            for key in keywords:
                if key in k['title'].lower():
                    url = shop_link + 'products/' + k['handle']
                    if url not in a:
                        a.append(url)
```

В зависимости от полученного результата делается вывод о том, доступен товар для покупки или ещё нет. В случае доступности пользователь получает уведомление в мессенджер. Для этого используется библиотека “dhooks”.

```
def monitor(url, atc_links, atc_links2, atc_links3, atc_links4, shop_link, image1, price, title):
    hook = Webhook('https://discord.com/api/webhooks/787406350099152926/iJ3TRwJmP-HJxngQutAXk8Ds2VTN7M')
    embed = Embed(description=shop_link, color=0x1e0f3)
    embed.set_title(title=title, url=url)
    embed.add_field(name='Цена:', value=price + '$', inline=False)
    embed.add_field(name='ATC', value=atc_links)
    if atc_links2 != '':
        embed.add_field(name='ATC', value=atc_links2)
    if atc_links3 != '':
        embed.add_field(name='ATC', value=atc_links3)
    if atc_links4 != '':
        embed.add_field(name='ATC', value=atc_links4)
    embed.set_thumbnail(image1)
    embed.set_footer(text='Developed by DoodkaTeam | ' + str(datetime.datetime.now())[11:-1])
    hook.send(embed=embed)
```

Для возможности отслеживания статусов нескольких товаров применяется многопоточность при помощи библиотеки «Threading».

В итоге пользователь оповещается о поступившем товаре на продажу следующим уведомлением на рисунке №5.



Рис. 5, способ отображения пользовательского уведомления в «Discord»

Благодаря «монитору», можно моментально узнать о поступлении товара и сразу получить ссылку на него.

2.4 «Бот»

Данное специализированное ПО добавляет вместо пользователя желаемый товар с нужными характеристиками (цвет, размер и т.п.) в корзину, проходит все этапы оформления заказа и совершает платеж от имени авторизованного пользователя.

В проекте предусмотрена легкая настройка: заполнение личных данных, выбор торговой площадки для покупки, указание ключевых слов для фильтрации товаров по определённым критериям размера, цвета и т.п., выбор времени запуска автоматизированных задач. Работа бота была протестирована на торговой площадке бренда «Nike» - Nike.com, так как на данной площадке еженедельно поступают на реализацию ограниченные серии товаров, в основном, обувь, которую крайне тяжело купить из-за быстрой распродажи при высоком спросе.

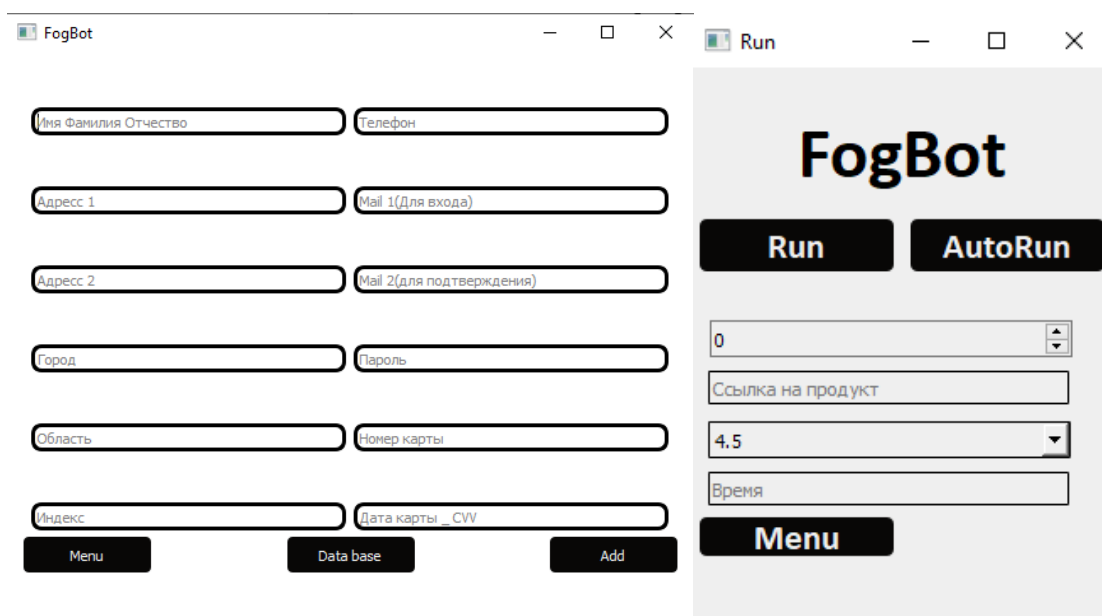


Рис. 6, интерфейсное окно «бота»

Принцип работы «бота» заключается в том, что после первичной настройки пользователем, заполнения всех персональных данных и указания авторизационных наборов от «авторегера», программное обеспечение за 30 минут до начала распродажи начинает свою активность в фоновых процессах веб-браузера на основе chromium: авторизуется за пользователя, открывает раннюю ссылку на товар (находит в коде сайта), заполняет все персональные данные и в нужное время активирует кнопку «отправить заказ».

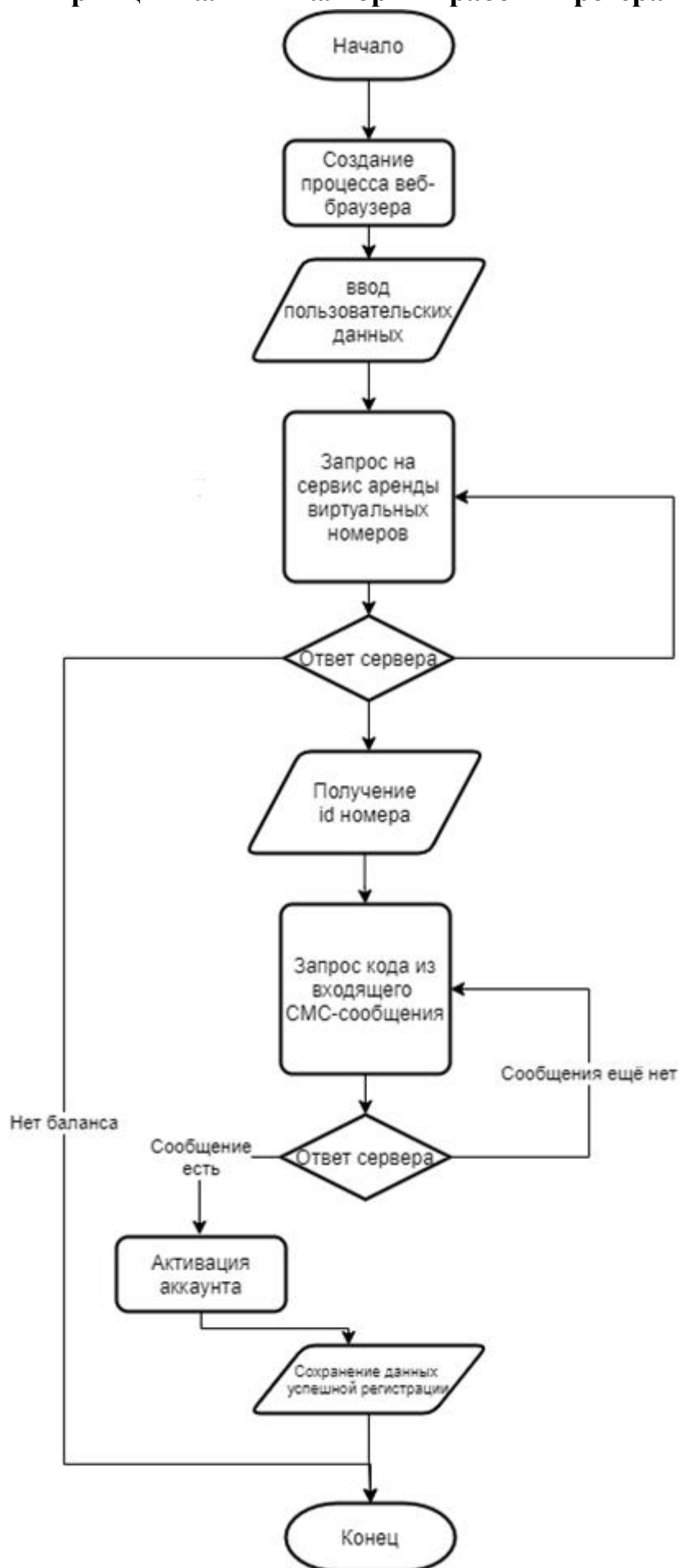
Для написания бота использована библиотека для python – «Selenium», благодаря которой можно контролировать поведение браузера на основе «chromium». В проекте использовался «Google chrome» из-за его распространённости на момент создания ПО.

Ниже приведён фрагмент кода программы, отвечающий за создание процесса веб-браузера с возможностью запуска как с прокси, так и без прокси. Обеспечивает взаимодействие библиотеки «Selenium» с процессами веб-браузера связующее ПО – «ChromeDriver». В библиотеке для открытия страницы по необходимой ссылке используется функция `get()`, в которую параметром передаётся нужный адрес на ресурс.

Список использованных источников

1. Бот программа. URL: [https://ru.wikipedia.org/wiki/Бот_\(программа\)](https://ru.wikipedia.org/wiki/Бот_(программа)) (дата обращения: 11.11.2020).
2. ChromeDriver. URL: <https://github.com/bayandin/chromedriver> (дата обращения: 24.11.2020)
3. Python requests. URL: <https://2.python-requests.org/en/master/user/advanced/> (дата обращения: 21.11.2020)
4. Discord WebHooks. URL: <https://dhooks.readthedocs.io/en/latest/> (дата обращения: 02.12.2020)
5. Руководство к Browser Automation Studio. URL: <https://wiki.bablosoft.com/doku.php?id=ru:start> (дата обращения: 15.11.2020)
6. Справочник по языку Python 3.1. URL: https://ru.wikibooks.org/wiki/Python/Справочник_по_языку_Python_3.1 (дата обращения: 18.11.2020)
7. JSON. URL: <https://ru.wikipedia.org/wiki/JSON#:~:text=JSON%20> (дата обращения 05.12.2020)
8. JSON encoder and decoder. URL: <https://docs.python.org/3/library/json.html> (дата обращения: 29.11.2020)

Приложение 1. Принципиальный алгоритм работы «регера»



Приложение 2.
Ход реализации проекта.

Февраль 2020	Постановка проблемы
Март – май 2020	Изучение специальной литературы
Июнь – август 2020	Проектирование системы
Сентябрь – октябрь 2020	Разработка «регера»
Ноябрь 2020	Разработка «монитора» и «бота»
Декабрь 2020	Тестирование совокупности ПО
2021	Привлечение потенциальных клиентов

