****

**ГБОУ Школа 1535 города Москвы**





**“Sandy Loader”. Браузерное расширение, выполняющее быструю, удобную и безопасную загрузку требуемых файлов со страницы сайта**

**Выполнил:**

**Ткачев Николай**

**ГБОУ «Школа 1535», 10В класс**

**Научный руководитель:**

**Жемчужников Д.Г., к.п.н.,**

**учитель информатики**

**Москва 2023**

**Оглавление**

[Команда проекта 3](#_heading=h.26in1rg)

[Аннотация 4](#_heading=h.35nkun2)

[Актуальность 5](#_heading=h.1ksv4uv)

[Новизна и анализ рынка 6](#_heading=h.2jxsxqh)

[Цели и задачи проекта 9](#_heading=h.qsh70q)

[План проекта 10](#_heading=h.49x2ik5)

[Ресурсы 12](#_heading=h.3o7alnk)

[Проектирование 13](#_heading=h.23ckvvd)

[Описание работы приложения 13](#_heading=h.ihv636)

[Разработка веб-скрипта 14](#_heading=h.32hioqz)

[Разработка расширения для Google Chrome 18](#_heading=h.1hmsyys)

[Серверная часть 18](#_heading=h.41mghml)

[Разработка оформления сайта 20](#_heading=h.vx1227)

[Сравнительный анализ решений 21](#_heading=h.3fwokq0)

[Результат, выводы и пути развития 22](#_heading=h.1v1yuxt)

[Модель и способ коммерциализации 22](#_heading=h.4f1mdlm)

[Пользовательские сценарии 23](#_heading=h.2u6wntf)

[Стратегия продвижения сервиса к потребителю 23](#_heading=h.19c6y18)

# Команда проекта

Участники проекта и их роли представлены в таблице 1.

Таблица 1. Команда проекта

| **ФИО** | **Должность** | **Функция в проекте** | **Задачи в проекте** |
| --- | --- | --- | --- |
| Жемчужников Дмитрий Григорьевич | Учитель информатики | Научный руководитель | 1. Организация работы проектной команды.  2. Обеспечение ресурсами. |
| Ткачев Николай | ГБОУ Школа №1535, 10 «В» | Веб-разработчик, создатель скрипта поиска, отладчик | 1. Разработка внешнего вида расширения и методов взаимодействия с пользователем. 2. Разработка серверной составляющей 3. Обучение скрипта 4. Подключение скрипта |
| Учащиеся 10В класса: 20 человек |  | Тестировщики | Тестирование расширения для браузера на выборке сайтов, включая "сомнительные” сайты |

# Аннотация

Рассмотрен **инновационный способ скачивания файлов**, который избавляет пользователей от долгого поиска ссылок и кнопок на странице сайта. Неопытные пользователи смогут получать нужные им объекты, проверенные через облачный антивирус, а опытные смогут быстрее получать требуемые данные. Это позволит сэкономить ценное время.

Посредством специального **расширения для браузера** пользователи смогут в два клика получить удобный список файлов со страницы сайта, который обновляется по мере поступления очередной ссылки.

Проектируемое расширение будет выдавать пользователю конкретные проверенные файлы, полученные путем **имитации нажатий на интерактивные части** страницы, что гарантирует получение требуемого файла, если он существует и проверен.

Программа будет независимо от пользователя производить действия с сайтом (переходить на другие страницы, скачивать, возможно, опасные файлы, принимать все соглашения и уведомления). Отсюда аналогия в названии с **песочницей**, где можно безопасно делать все, что захочется.

Ключевые слова: скачивание файлов, парсинг, браузерное расширение, мастер загрузок, захват файлов, безопасная загрузка, веб-бот.

# Актуальность

Потребность в **эффективном и безопасном скачивании** всегда существовала среди всех пользователей интернета, которые когда-либо сталкивались с поиском и закачкой файлов.

**1.** Люди, которые не умеют пользоваться интернетом, могут часами искать нужную информацию из-за недостаточного опыта. Дети просят своих родителей скачать им компьютерную игру, бабушки и дедушки просят своих сыновей и внуков скачать книгу, инструкцию или фильм.

**2.** Иногда поиск нужного файла даже для опытного пользователя занимает много времени: его постоянно перекидывает **на сторонние страницы с рекламными предложениями** и просят включить уведомления, разрешить доступ или подождать какое-то количество времени (рис. 1). Блокировщик рекламы (Adblock) не уменьшает время поиска информации и опасность получения вируса, а лишь убирает нежелательную рекламу.

**3.** Антивирус очень часто реализован неудобно и **удаляет скачанные файлы**. Во избежание этого, нужно настраивать определенный антивирус, а это довольно сложно и долго.

Эти проблемы были бы решены, если бы существовал такой **бот-помощник, который избавил пользователей от долгого поиска файлов, защитил от вредоносных программ и рекламы и был удобным и понятным для каждого.**

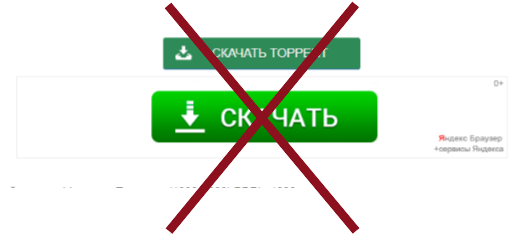


Рисунок 1. Нежелательные ссылки, которых остерегаются пользователи

# Новизна и анализ рынка

Первичный анализ рынка и поиск аналогов в сети интернет показал, что не существует сайтов или расширений для браузеров, способных корректно получать файлы со страницы сайта, проверять их и выдавать пользователю в приемлемом виде.

Существует несколько реализаций этой идеи (скачивания файлов), но все они имеют существенные недостатки и никак не связаны с основной чертой этого проекта: имитацией действий пользователя.

1. **Download Master (рис. 2)**

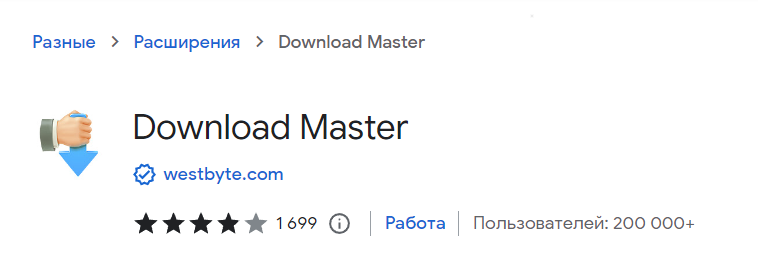


Рисунок 2. Страница Download Master в магазине расширений Chrome

Расширение Download Master [1] работает исключительно с помощью своего приложения, которое надо скачивать отдельно и настраивать. Это приложение предназначено для того, чтобы получить все элементы со страницы сайта. Удобно для парсинга, но для закачки файлов - нет. Download Master находит **только** **ссылки** на одной странице сайта (картинки, другие страницы, возможные ссылки на файлы). Вместе с приложением Download Master получает все эти элементы за 3-5 секунд (рис. 3). Список становится огромным и понятным только для профессионала.

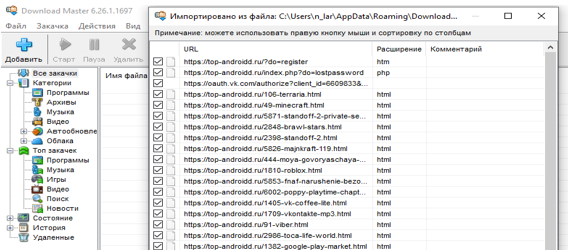


Рисунок 3. Результат работы Download Master

1. **Free Download Manager (рис. 4)**

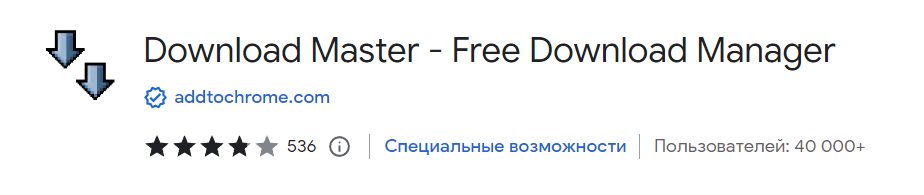


Рисунок 4. Страница Download Master – Free Download Manager в магазине расширений Chrome

Free Download Manager [2] работает примерно по такому же принципу, что и предыдущее, но **не использует свое приложение**.

Обе реализации, безусловно, работают быстро, но только из-за того, что осуществляют поиск по внутренней разметке сайта, следовательно, показывают пользователю какие-то **непонятные ссылки с непонятным названием** (рис. 5).

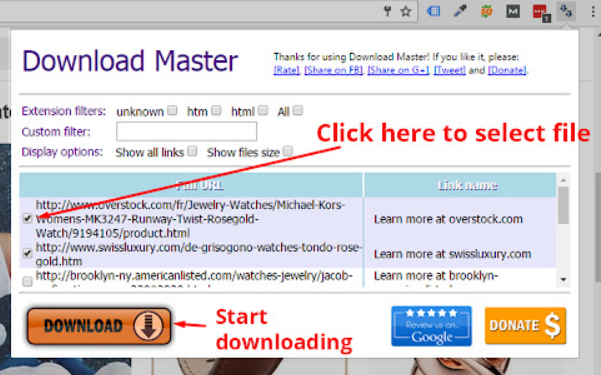


Рисунок 5. Результат работы Download Master – Free Download Manager

Можно заметить, что эти расширения очень востребованы среди пользователей (200.000+ скачиваний у Download Master). С учетом планируемых показателей, разрабатываемое расширение не уступает вышеперечисленным. Можно предполагать, что оно будет востребованным.

# Цели и задачи проекта

**Проблема**

**Отсутствие на рынке инструмента для браузера, способного обходить фейковые ссылки на файлы (в любом виде) и безопасно получать требуемые файлы со страницы сайта.**

**Цель проекта**

**Создать удобное расширение для браузера с лаконичным и дружественным интерфейсом, которое будет осуществлять эффективный поиск требуемых файлов.**

Цель определила **задачи проекта:**

* + - 1. Изучить основы Python, Selenium, Django, JavaScript, HTML, CSS
      2. Создать расширение для браузера
      3. Создать и обучить скрипт, способный выполнять поиск файлов и интерактивные операции на странице
      4. Создать сервер на Django, подключить к нему скрипт
      5. Протестировать расширение
      6. Разместить расширение в сети интернет, в интернет-магазине расширений Chrome
      7. Апробировать расширение среди пользователей Google Chrome
      8. Отладить и доработать код, получив обратную связь

**Потенциальные потребители:** пользователи интернета, которые хотят скачать какие-либо файлы.

**Планируемые результаты:**

* обнаружение и проверка всех требуемых файлов на странице не более чем за 1 минуту и с точностью от 85%;
* успешное тестирование не менее 200 веб-страниц;
* апробация не менее чем 20 пользователями.

# План проекта

Начало: 1.09.2022 г. Окончание: 1.03.2023 г.

План проекта представлен в таблице 2 и рис. 6. Условные обозначения ролей участников: Т – Ткачев Николай, НР – научный руководитель

Таблица 2. Календарный план проекта

| **№ п/п** | **Название этапа** | **Сроки реализации** | **Участники** |
| --- | --- | --- | --- |
| **1** | **Подготовка, планирование** |  |  |
| 1.1 | Постановка цели, обсуждение основной концепции и технических особенностей проекта | 1.09 – 20.09 | Т, НР |
| 1.2 | Выбор инструментов разработки | 20.09 – 15.10 | Т, НР |
| **2** | **Разработка** |  |  |
| 2.1 | Обсуждение деталей реализации | 1.10 – 30.10 | Т, НР |
| 2.2 | Разработка прототипа сайта без бэкенда | 30.10 – 15.11 | Т |
| 2.3 | Изучение литературы | 1.10 – 31.12 | Т |
| 2.4 | Программирование основного алгоритма поиска файлов (selenium) | 15.11 – 31.12 | Т |
| 2.5 | Написание бэкенда | 5.12 – 20.12 | Т |
| 2.6 | Внедрение в качестве расширения для браузера Google Chrome | 20.12 – 15.01 | Т |
| 2.7 | Тестирование различных вариантов реализации проекта | 20.12 – 20.01 | Т |
| 2.8 | Разработка финальной версия дизайна расширения | 1.01 – 20.01 | Т |
| 2.9 | Калибровка скрипта | 1.01 – 10.01 | Т |
| 2.10 | Совмещение фронтенда и бэкенда | 10.01 –31.01 | Т |
| 2.11 | Финальное тестирование, внесение изменений | 20.01 – 31.01 | Т |
| **3** | **Подведение итогов, развитие** |  |  |
| 3.1 | Документация и оформление проекта | 1.02 –28.02 | Т, НР |
| 3.2 | Создание демонстрационных материалов | 10.02 –28.02 | Т, НР |

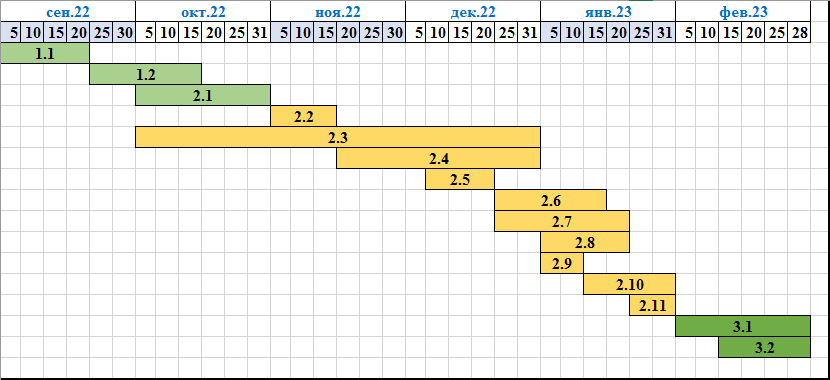
****

Рисунок 6. Дорожная карта проекта (диаграмма Ганта)

# Ресурсы

* Подключение к сети Интернет (~ 500 руб. / мес.)
* Компьютерное ПО - база проекта
* Процессор AMD Ryzen 5 3600 6-Core (использовалась для быстрого запуска и тестирования скрипта) - из личного пользования
* Рабочее и личное время разработчика - 100 часов
* Личное время тестировщиков - совокупно 10 часов
* Рабочее и личное время научного руководителя - 10 часов

# Проектирование

## Описание работы приложения

Порядок работы в приложении можно разбить на три этапа:

1. Пользователь заходит на сайт
2. Нажимает на любую свободную часть страницы правой кнопкой мыши, в контекстном меню выбирает расширение Sandy Loader (рис. 7)

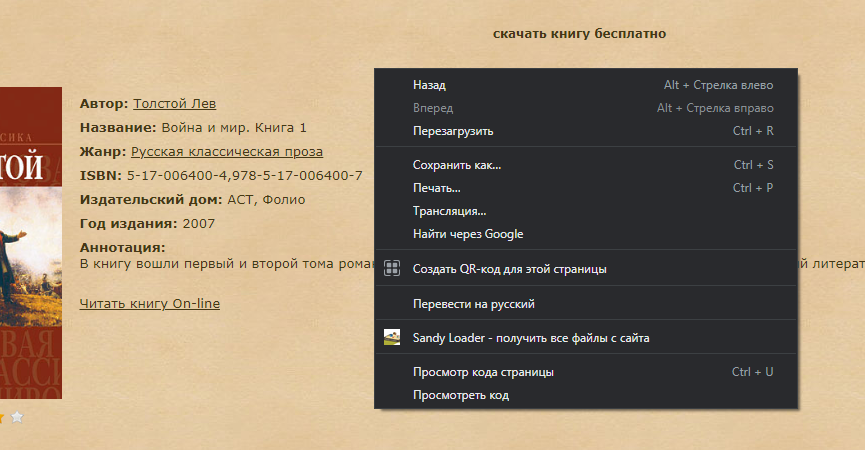


Рисунок 7. Изображение работы приложения

1. Расширение отправляет запрос на сервер, переадресовывает пользователя на сайт Sandy Loader
2. Сайт показывает пользователю все найденные файлы с данной страницы, указывает их степень опасности (рис. 8)

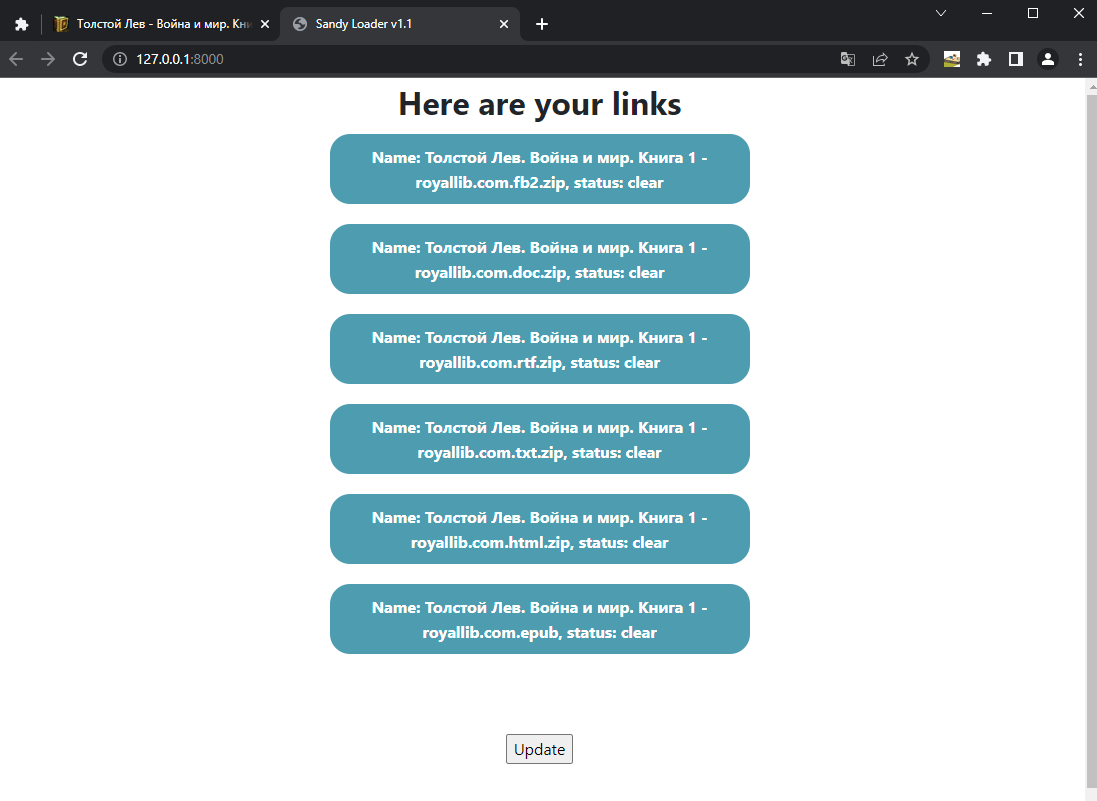


Рисунок 8. Результат работы приложения

## Разработка веб-скрипта

Решено было начать с разработки скрипта поиска файлов на странице. Для начала требовалось понимание работы веб языков и языка **Selenium** в частности. После просмотра десятков обучающих роликов и прочтения документации появилась возможность получать знания на практике.

На рис. 9 изображена предварительная концепция работы всей программы.

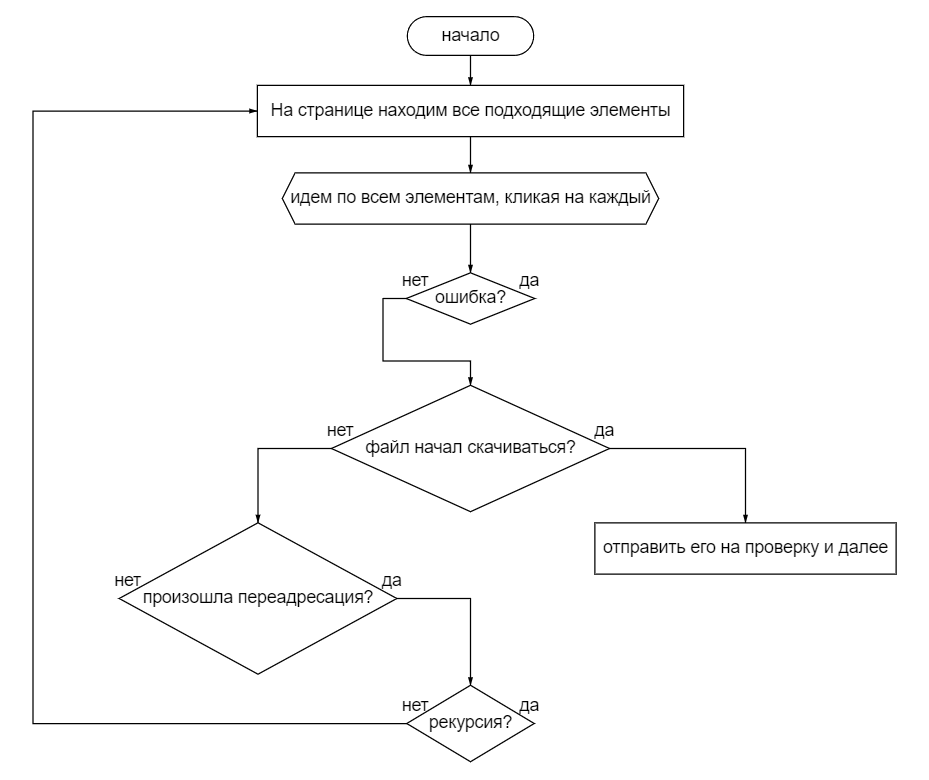


Рисунок 9. Концепция работы программы (упрощенная схема)

Для начала надо было понять, как со страницы выбрать все элементы и затем отсортировать их. Было принято решение выполнять поиск по тегам “<a>”, “<p>”, “<span>” и “<button>”, потому что, как правило, все элементы, связанные с загрузкой, находятся в них. Так как программа должна работать достаточно быстро, пришлось игнорировать некоторые, возможно подходящие, элементы. Чтобы быстро получить все атрибуты из элемента написана функция на С++ и преобразована с помощью **execute\_script** для запуска ее в основном скрипте на Python.

Первой крупной проблемой оказалась работа с разметкой сайтов и DOM (Document Object Model). Самой распространенной ошибкой была “Element is no longer attached to the DOM” при клике на элемент. Эта ошибка была вызвана тем, что **сайты часто переадресовывают пользователя на другие страницы или блокируют доступ к элементу**. В отличие от реального пользователя, бот не видит элементов страницы на экране, а гуляет по DOM, как по лабиринту. Соответственно пришлось добавить некоторые исключения и в корне изменить логику скрипта (которая в дальнейшем десятки раз переписывалась).

Следующей задачей было получить названия скачанных файлов. Для этого использовалась библиотека **os** (для доступа к папкам в системе). В настройках браузера (при инициализации в скрипте) понадобилось указать путь, куда будут скачиваться все файлы. После изучения функций библиотеки сформировалась программа (рис. 10):



Рисунок 10. Часть программы, отвечающая за скачивание файлов

После того, как элемент прошел все проверки, программа открывает папку загрузки и проверяет ее. Если она не пустая (если файл скачался либо файл еще скачивается вместе с хеш файлом), то ссылка из элемента посылается на проверку в антивирус вместе с названием файла. Далее, если файл еще скачивается, то попытка его удалить вызовет ошибку, поэтому в самом браузере нужно отменить загрузку. В ином случае мы просто удаляем все содержимое папки. Все эти действия наглядно продемонстрированы на рисунке 11.

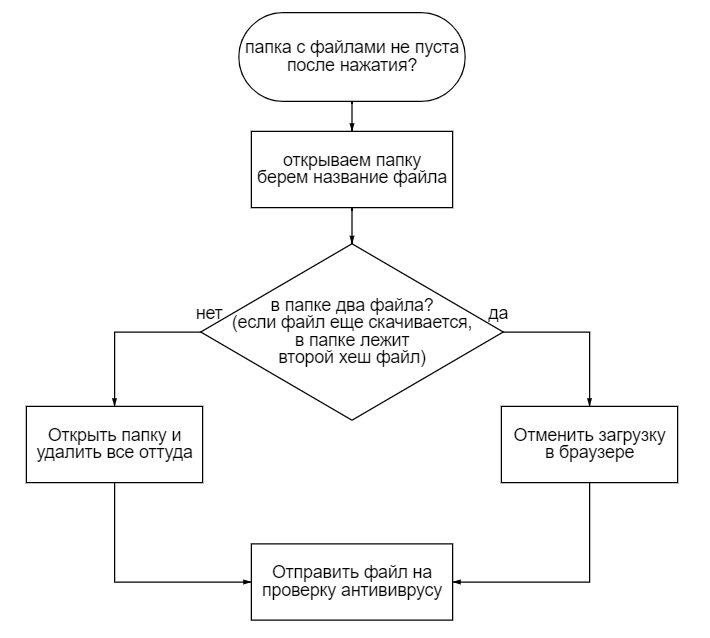


Рисунок 11. Концепция проверки получения файла

Все действия скрипта выполняются на странице сайта, которую указал пользователь (ссылка от расширения). К сожалению, гарантировать получения файла таким образом нельзя, поэтому пришлось **доработать программу**, чтобы она смогла выполнять поиск и на других страницах (если произошла переадресация при нажатии на элемент). **Рекурсия была ограничена до 1 перехода на каждый элемент** (если файла нет на второй странице, то возвращаемся на первую). После этих изменений программа стала очень долго работать. Понадобилось переписать вызов функции и уменьшить список ключевых слов поиска.

Для проверки файлов был использован бесплатный веб-антивирус у [Dr.Web](https://vms.drweb.ru/online/). Функция антивируса в основном файле скрипта принимает значения от функции поиска и посылает ссылку антивирусу (рис. 12). Программа снова начала работать очень медленно. Пришлось сделать **две оптимизации**:

1. **Если проверка длится более 4 секунд, то переходим к следующему файлу.**
2. **Для поиска и для проверки создаем два разных веб-драйвера. При запуске сервера инициализируем их сразу.**

В итоге антивирус успешно проверяет файлы до 15 Гб (этого вполне достаточно).



Рисунок 12. Исходный код процесса передачи файла антивирусу

## Разработка расширения для Google Chrome

Чтобы максимально упростить передачу логина текущей страницы на сайт, будем разрабатывать расширение для Google Chrome. Для его создания нам требуются 3 файла.

* **manifest.json** — требуемая браузером информация о расширении
* **eventPage.js** — сам скрипт, передающий ссылку, откуда было вызвано расширение. Он подает ссылку через запрос GET на сервер Jango. Сервер в это время передает ссылку уже запущенному веб-драйверу, который выполняет поиск файлов и отправляет его на сайт.
* **popup.html** — краткая информация о расширении

В процессе разработки выяснилось, что для полноценной работы расширения требуется разрешения tabs и context Menus, которые необходимо объявить в файле manifest.json. Разрешение “tabs” предоставляет доступ к открытым вкладкам в браузере, а разрешение “contextMenus” доступ к контекстному меню. В это меню был добавлен пункт "Sandy Loader - получить все файлы с сайта", но он появляется только в том случае, если пользователь кликнул именно на свободную часть страницы (не картинка, не ссылка, не текст). Если действие будет выполнено успешно, то будет выполнена основная onclick-функция, которая создает в браузере вкладку с сайтом Sandy Loader.

## Серверная часть

Разработка серверной части началась с установки фреймворка Django. Через консоль было установлено приложение taskmanager, в нем находятся все необходимые настройки и скрипт поиска. Работа затянулась на моменте обработки принимаемого значения из расширения. Расширение отправляло запрос GET на сервер, но сервер его не принимал. Оказалось, что в настройках сервера нужно поместить адрес расширения в разрешенные адреса, а затем принимать запросы в **views.py** (который отвечает за обработку запросов). Далее пришлось неоднократно дорабатывать файл views.py, чтобы установить связь между скриптом и запросом от расширения (рис. 13).

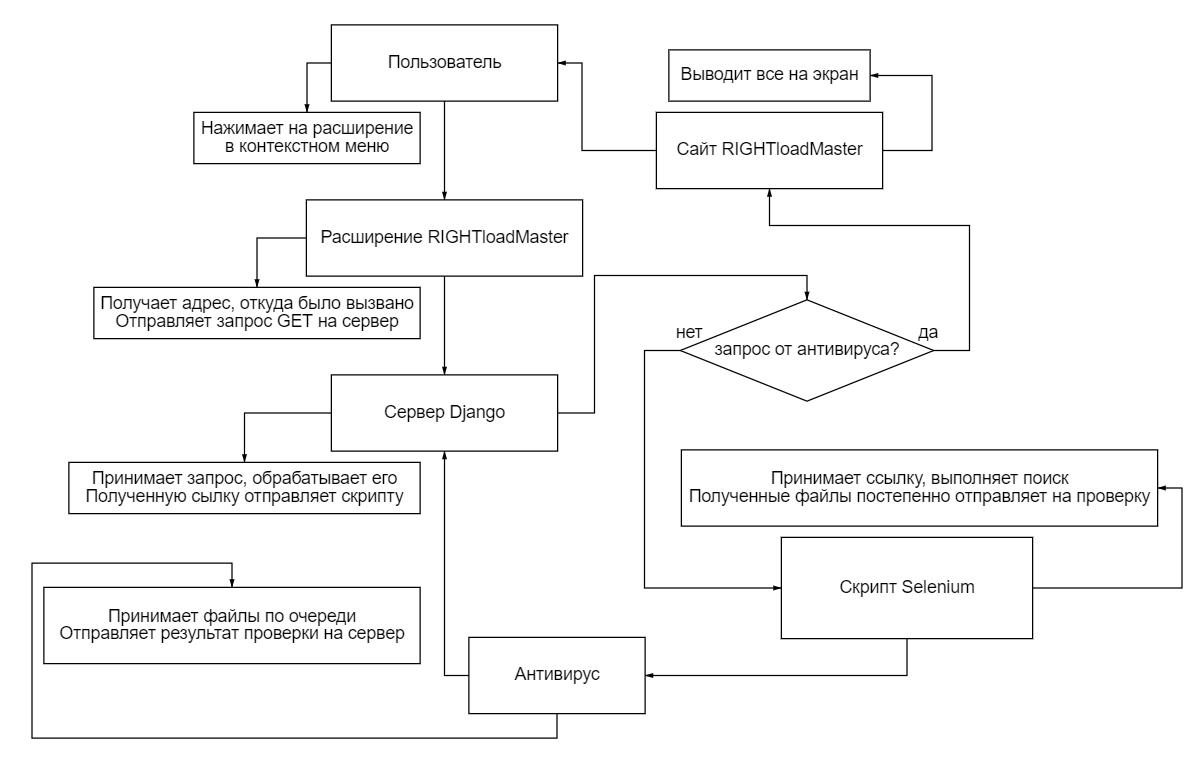


Рисунок 13. Концепт серверной системы

В процессе реализации понадобилось немного изменить систему. Вместо запроса на сервер, антивирус записывает результаты в файл (рис. 14). Кнопка обновления на сайте подключена к форме, которая при нажатии автоматически отправляет POST на сервер. Далее сайт обновляется, новые ссылки на файлы переписываются.

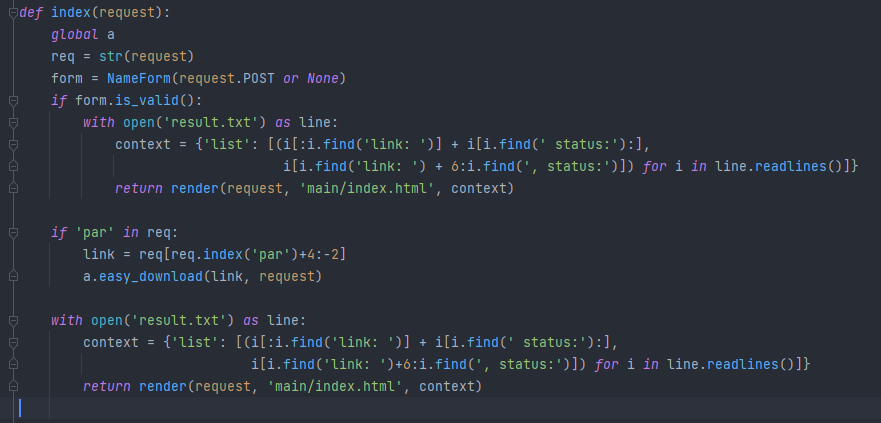


Рисунок 14. Исходный код функции, принимающей запросы (views.py)

## Разработка оформления сайта

Задача состоит в том, чтобы сделать минималистичный и интуитивно понятный интерфейс. На сайте будут выводиться названия скачанных файлов, статусы проверки и ссылки на эти файлы.

Для вывода названий файлов и ссылок пришлось изменить серверную часть. Теперь результаты скачивания записываются в файл, а затем считываются в разметке на сервере (рис. 15).

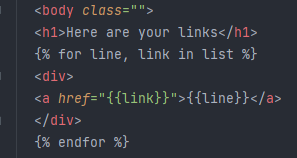


Рисунок 15. Вывод ссылок на сайт через разметку

## Тестирование

Тестирование программы было проведено на более чем 250 сайтах 15 учащимися 10В класса. Каждому тестировщику на компьютер было установлено расширение Sandy Loader. Каждый учащийся перешел на 16 сайтов в Google Chrome и вызвал расширение.

# Сравнительный анализ решений

Сравнение расширения Sandy Loader с существующими решениями по критериям представлено в таблице 3.

Таблица 3. Сравнение с аналогами (по задаче скачивания) по 10-ти балльной шкале

|  | Download Master - Free Download Manager | Download Master | **Sandy Loader** |
| --- | --- | --- | --- |
| Удобство | 6 | 7 | 10 |
| Веб-версия | + | - | + |
| Языки (кол-во) | 1 | Более двух | Более двух |
| Качество вывода информации пользователю | 4 | 6 | 10 |
| Быстрота | 9 | 10 | 6 |
| Проверка на вирусы | - | - | + |
| Эффективность поиска файлов | 2 | 4 | 10 |
| Итого | 22 | 27 | **36** |

# Результат, выводы и пути развития

Поставленная **цель достигнута** в том виде, в котором планировалась. Проект завершен.

Изучены основы Python, Selenium, Django, JavaScript, HTML, CSS. Благодаря самообучению и обучению на курсе Лицея Академии Яндекса “Основы промышленного программирования”

Разработано **полнофункциональное и действующее расширение** для Google Chrome, позволяющее эффективно получать файлы со страницы сайта и выдавать пользователю в безопасном виде.

Создан **полнофункциональный сервер на Django**, выполняющий обмен данными между пользователем и скриптом.

Программа находит, проверяет и выводит пользователю первый файл за **10-15 секунд**. За **40 секунд** программа находит и проверяет уже **более 4 файлов весом до 50 гигабайт**.

Тестирование все еще проводится, и текущая вероятность получения требуемого файла со страницы **составляет 96%.**

Расширение протестировано на **более чем 250 сайтах**, в тестировании принимали участие одноклассники (15 человек).

**Целевые показатели проекта достигнуты.**

Расширение пока **не размещено** в магазине расширений Chrome (нет средств для покупки аккаунта разработчика расширений Chrome)

Расширение пока **не апробировано** среди пользователей Google Chrome, т.к. оно не было размещено в магазине расширений.

**Результат работы приложения можно посмотреть по** [**ссылке**](https://drive.google.com/file/d/1ay9bmGLPhYmqCZ1WvyNEaKsVE8VpBSuQ/view?usp=share_link)

**Исходный код программы (github)** [**здесь**](https://github.com/NLarinov/Sandy-Loader.git)

Рассмотрим пути развития проекта.

## Модель и способ коммерциализации

Для данного проекта, как способ коммерциализации, было выбрано самостоятельное выведение на рынок, без передачи прав на инновацию. Для дальнейшего развития проекта на рынке сформировалась следующая модель (рис. 16).

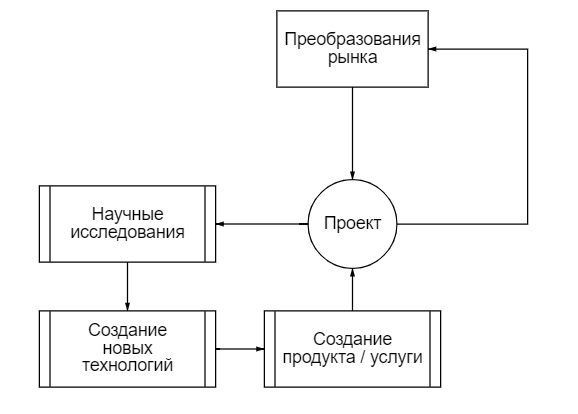


Рисунок 16. Модель коммерциализации

## Пользовательские сценарии

Проведя опрос, удалось выделить четыре наиболее распространенные категории пользователей:

1. Люди любого возраста, которые хотят скачать какие-либо развлекательные материалы (фильмы, сериалы, игры, музыка и т.д.), но не хотят тратить время на поиск, закачку и проверку файлов.
2. Школьники, студенты, учителя и профессора, у которых часто возникают трудности в скачивании обучающих материалов (учебников, презентаций, аудиофайлов и т.д.).
3. Пользователи, плохо ориентирующиеся в интернете (как правило дети или старики), которые не могут найти файл на странице либо боятся это делать. Чаще всего такие пользователи просили помощи у своих знакомых или родственников, но такая помощь предоставляется редко и далеко не всем.
4. Пользователи, опасающиеся за безопасность своего устройства. Обычно такие люди устанавливают антивирус себе на устройство, но многих раздражает, то, что антивирусы начинают хозяйничать на устройстве и требуют плату за свои услуги.

## Стратегия продвижения сервиса к потребителю

Сначала планируется **выгрузить** сервер на хостинг и **настроить** его для большого количества запросов. Далее **протестировать** приложение в масштабах школы. Такой пробный запуск поможет выявить недостатки.

Проанализировав результаты и внеся изменения в приложение, планируется **начать рекламировать** расширение на различных платформах и форумах.

**Результат работы приложения можно посмотреть по** [**ссылке**](https://drive.google.com/file/d/1ay9bmGLPhYmqCZ1WvyNEaKsVE8VpBSuQ/view?usp=share_link)

**Исходный код программы (github)** [**здесь**](https://github.com/NLarinov/Sandy-Loader.git)

## Список источников и использованной литературы

1. Страница Download Master в магазине расширений Chrome [Электронный ресурс]. URL: [https://chrome.google.com/webstore/detail/download-master/ dljdacfojgikogldjffnkdcielnklkce?hl=ru](https://chrome.google.com/webstore/detail/download-master/dljdacfojgikogldjffnkdcielnklkce?hl=ru)
2. Страница Free Download Manager в магазине расширений Chrome [Электронный ресурс]. URL: [https://chrome.google.com/webstore/detail/download-master-free-down/ laepcndcehndnjndpfjdcdgbneoimdgg?hl=ru](https://chrome.google.com/webstore/detail/download-master-free-down/laepcndcehndnjndpfjdcdgbneoimdgg?hl=ru)
3. Selenium Webdriver [Электронный ресурс]. URL: <https://wiki.dmdevelopment.ru/wiki/Download/Books/development/selenium-webdriver.pdf>
4. Selenium Python Built [Электронный ресурс]. URL: <https://selenium-python.readthedocs.io/_/downloads/en/latest/pdf/>
5. Test-Driven Web Development with Python [Электронный ресурс]. URL: <https://www.obeythetestinggoat.com/pages/book.html>
6. Creating Google Chrome Extensions 1st ed. Edition by Prateek Mehta [Электронный ресурс]. URL: <https://link.springer.com/content/pdf/bfm:978-1-4842-1775-7/1.pdf>
7. Stackoverflow [Электронный ресурс]. URL: <https://stackoverflow.com/>
8. Habr [Электронный ресурс]. URL: <https://habr.com/>
9. Cyberforum [Электронный ресурс]. URL: <https://www.cyberforum.ru/>