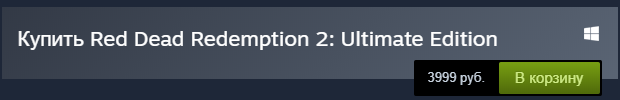
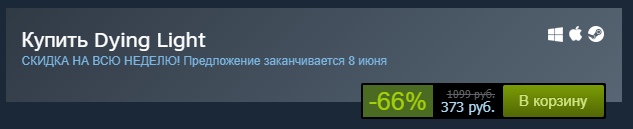
|  |  |  |
| --- | --- | --- |
| ДЕПАРТАМЕНТ ОБРАЗОВАНИЯ ГОРОДА МОСКВЫ  Юго-Западный административный округ  Государственное бюджетное общеобразовательное учреждение города Москвы **«Школа № 1533 «ЛИТ»** | | |
| **ВЫПУСКНОЙ ПРОЕКТ** | | |
|  | Учащихся  Гулякина Андрея Алексеевича  и  Зеленова Даниила Андреевича  Группы 10.4 |  |
| **“Создание бота-помощника для автоматизации работы с торговой площадкой Steam”**  Консультант – Борзых Денис Александрович | | |

Оглавление

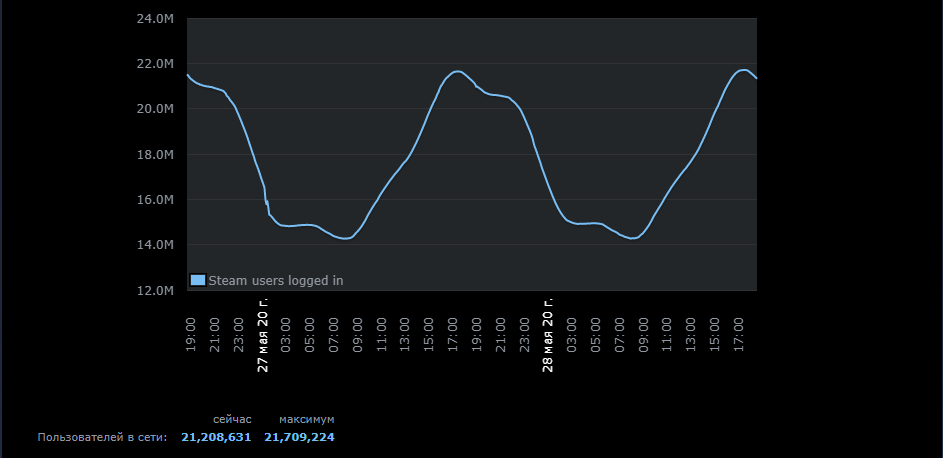
Введение

Ежедневно мы покупаем различные продукты для полноценного и спокойного жизнеобеспечения. Мы покупаем продукты, предметы гигиены, игрушки детям, корм питомцам -- в наши дни любой товар стоит каких-то денег. Всем приятно найти нужный товар по скидке и сэкономить на этом даже небольшую сумму. Роль скидок в нашем мире уже давно всем понятна. Теперь окунемся в реалии игровой индустрии и различных видеоигр. Уже несколько десятков лет игровая индустрия развивается и становится все популярнее по всему миру. За развитием игровой индустрии следует и развитие монетизации в ней. Например, в современных реалиях одна из лучших игр 2019 года на самой популярной площадке игр на данный момент стоит 4000 рублей.

Скидки присутствуют и в игровой индустрии.



Но на платных играх весь цифровой контент не ограничивается. В различных играх есть возможная кастомизация, за котороую разработчики требуют свою плату. Так же, Steam(самая популярная платформа распространения игр и программ, контролируемая компанией Valve во главе с Гейбом Ньюэлл) предоставила возможность обычным пользователям продавать различный цифровой контент за свою установленную цену, некий рынок. В основном это различные раскраски или не влияющие на игровой процесс предметы внешнего вида, которые позволяют отличиться даже в игре.

Что произошло бы, если о скидках в магазинах нас бы оповещал специальный помощник? А если бы этот помощник покупал нужные товары со скидкой? Экономия сразу видна, и никто бы не отказался от такой приятной вещи. И наш проект заключается в реализации этого помощника(бота) в интернете, для работы с торговой площадкой Steam(далее ТП). Миллионы игроков и пользователей ежедневно тратят деньги на ТП, покупая различную визуальную кастомизацию для своей любимой игры. Наша задача – упростить этот процесс и сделать его выгодным для обычного пользователя ТП. Актуальность этого проекта заключается в том, что ежедневно Steam-ом и его ТП пользуется больше 10 миллионов людей.

Так же у проекта нет прямых аналогов, у которых будут такие же возможности, как и у бота. Так же во время карантина количество геймеров выросло, из чего следует и рост спроса на различную визуальную составляющих игр.

Постановка задачи

В первую очередь, была поставлена задача с определением дизайна проекта, приятным для глаза пользователя. Было решено создать сайт для удобного и понятного взаимодействия с ботом. После написания визуальной оболочки сайта появилась группа задач по реализации взаимодействия пользователя с ботом через сайт. Группа состояла из н задач. Первая задача – реализация поиска предмета в базе данных сайта. Эта задача, в свою очередь, разделилась ещё на несколько: написание скрипта для перевода базы данных из исходного формата(.json) в текст и обратно, а так же реализация алгоритма по поиску нужного имени предмета в текстовом виде базы данных по введённому пользователем тексту в поисковую строку сайта. Вторая задача – реализовать грамотный графический вывод всех нужных пользователю предметов с дополнительными возможностями (купить, или зайти на страницу предмета в торговой площадке Steam). Далее в очереди стояла задача с созданием базы данных предметов, доступных боту для покупки. После этого необходимо было создать аккаунт бота в системе пользователей Steam для возможности покупать предметы с торговой площадки, а также отправлять эти предметы пользователям. Далее стояла задача с реализацией главной задачи проекта – возможностью купить предмет, когда его стоимость опустится до той, которую указал пользователь. В планах так же была реализация авторизация пользователя через его аккаунт в Steam, однако в финальную версию проекта она не попала из-за своей чрезмерной сложности в эксплуатации и малой пользой при применении.

Целевая аудитория и аналоги

Из аналогов можно выделить 3 штуки: Xbotapp, SteamMarketBot, Steam Trader Helper.

Xbotapp

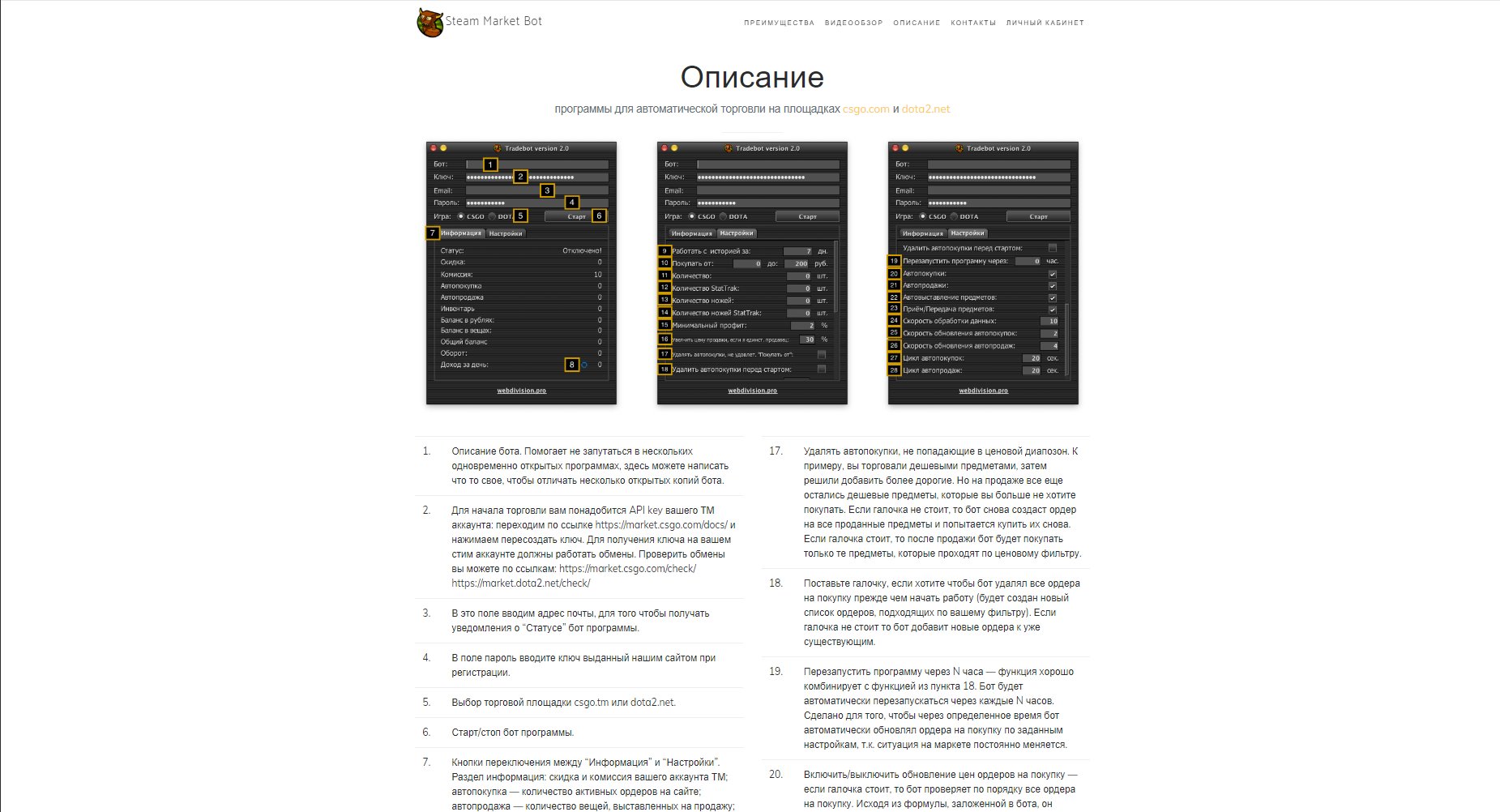
XBOT - бот для полной автоматизации торговли и заработка на площадках [market.csgo.com](https://vk.com/away.php?to=http%3A%2F%2Fmarket.csgo.com&cc_key=)\ [market.dota2.net](https://vk.com/away.php?to=http%3A%2F%2Fmarket.dota2.net&cc_key=). XBOT - программа на Windows 8\10, позволяющая полностью автоматически зарабатывать на торговле скинами, принимать и передавать вещи, или же просто продавать ваши скины на площадках [market.csgo.com](https://vk.com/away.php?to=http%3A%2F%2Fmarket.csgo.com&cc_key=)\[market.dota2.net](https://vk.com/away.php?to=http%3A%2F%2Fmarket.dota2.net&cc_key=). Плюсы этого проекта - удобность и скорость работы, но так как данный проект небесплатен (не менее 500р/месяц) и для его полного анализа потребовалось бы его покупать, описание проекта составлялось по отзывам из интернета.

Скриншот:



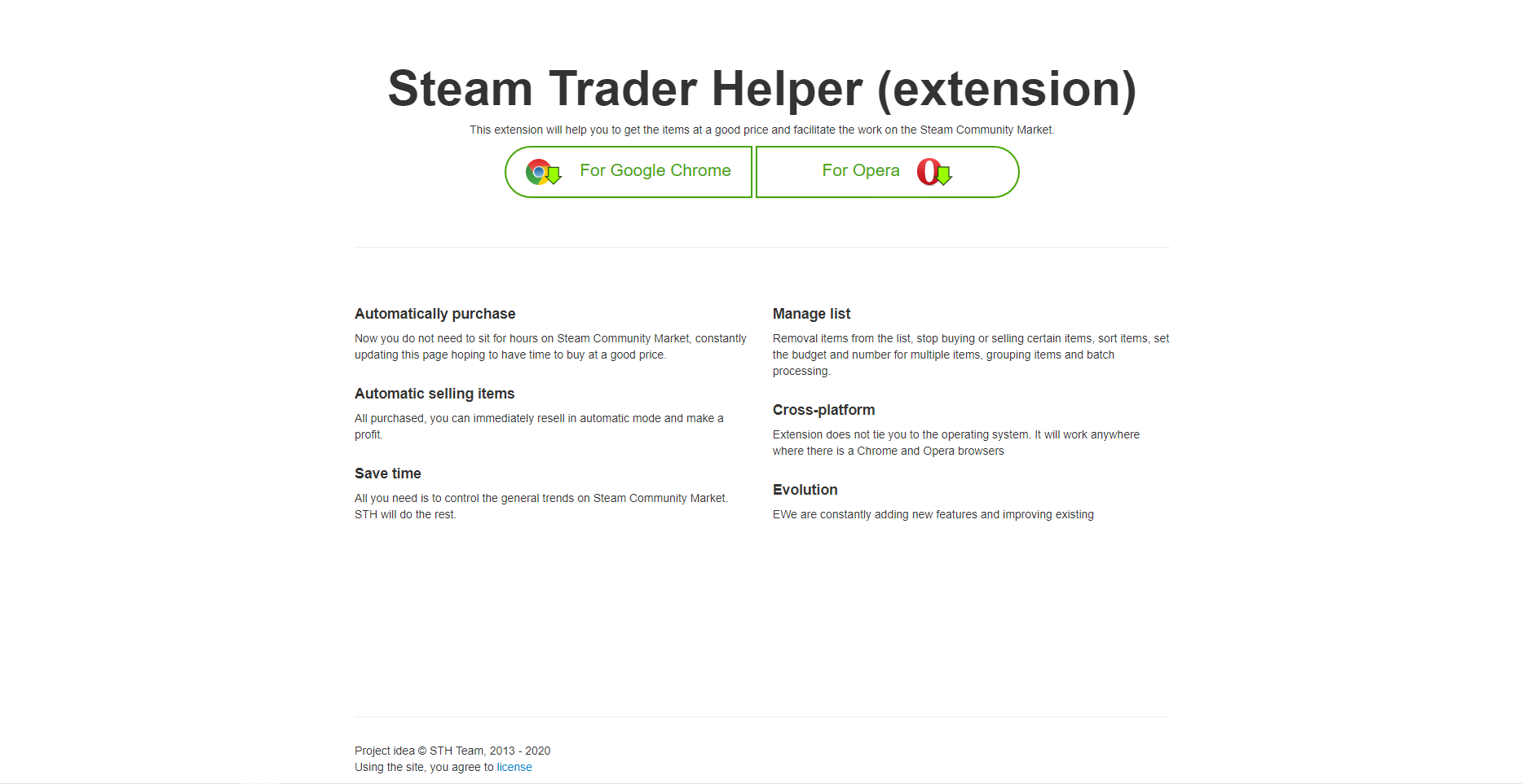
Steam market bot

Steam market bot - программа для автоматической торговли на площадках [csgo.com](https://market.csgo.com/), [dota2.net](https://market.dota2.net/), market.pubg.com и tf2.tm, которые, опять же, не являются оригинальными торговыми площадками Steam, в чём и состоит главный минус данного аналога. Из плюсов – широкий спектр торговых площадок, с которым работает проект, а также полная автономность и большое количество дополнительных функций. Скриншот:



Steam Trader Helper

Steam Trader Helper - расширение для Google Chrome и Opera. Оно служит для удобства и автоматизации торговли на торговой площадке Steam. Его главные положительные черты – бесплатность(расширение) и возможность не скачивать сторонние приложения, что обеспечивает безопасность данных пользователя. Из минусов - непонятный интерфейс, а так же то - что эта программа - лишь расширение для Chrome/Opera, что означает полную неподдерживаемость в остальных браузерах. Скриншот:



Начальная целевая аудитория проекта – друзья и знакомые исполнителей. Обусловлена такая узконаправленность только невозможностью грамотного пополнения баланса для покупок на данный момент. В дальнейшем будущем планируется расширение целевой аудитории до группы людей, осведомлённых о подобных программах и людей, интересующихся в торговле на торговой площадке Steam.

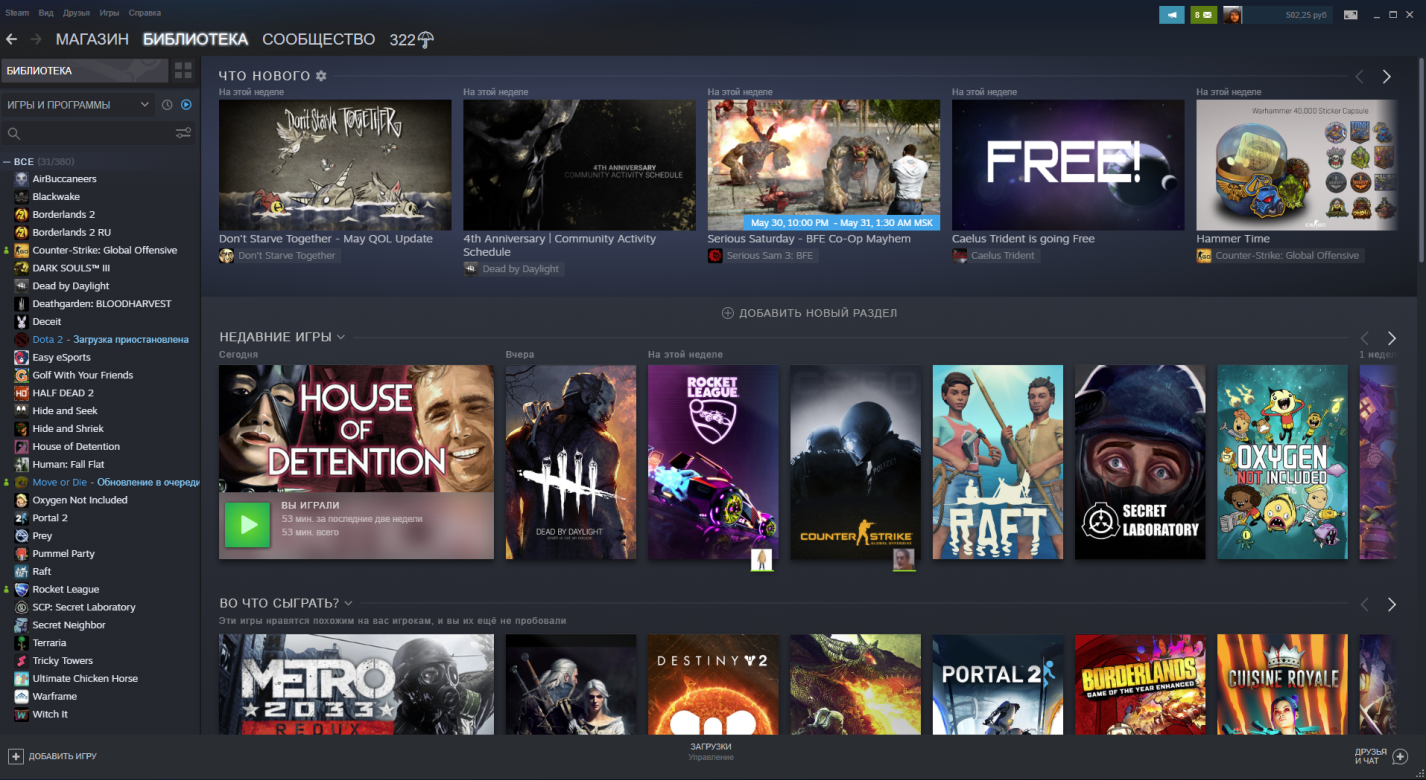
Решение

Ход работы

Изначально планировалось писать проект на языке программирования C#, то есть, написать оконное приложение на подобии аналогов данного проекта. После нескольких дней с начала работы над проектом, появились вопросы к дизайну проекта. Стало понятно, что при написании проекта на C# появятся проблемы с дизайном. Было решено сделать дизайн, приятный для глаз пользователя и так же удобный в использовании и понятный. После пары подсказок от одноклассников, хорошо разбирающихся в дизайне, всё пришло к выводу о том, что для хорошего дизайна нужно делать сайт. Для этого необходимо было ознакомиться с материалом по языку верстки и написания сайтов HTML, а так же для написания скриптов для бота необходимо было ознакомиться с материалом по языку, больше всех совместимому с сайтами – JavaScript, а так же с инструментами, облегчающими работу с Web-приложениями, таких как NodeJS(для облегчения запуска Web-приложений) и ExpressJS(для облегчения работы с get-запросами на сервер). Так же по рекомендации одного близкого знакомого, тесно связанного с дизайнерством сайтов было принято решение облегчить написание внешней оболочки сайта с дополнительным использованием фреймворка Vuetify. С его помощью, получилась довольно красивая и в то же время простая внешняя оболочка сайта. После этого началось изучение JavaScript в рамках функций сайта, нужных для проекта (написание обработки get-запроса пользователя к серверу, а так же скриптов, связанных с базой данных). После решения проблемы с внешним видом проекта, надо было решить проблему с внутренним составляющим проекта. Почти сразу было решено, что для удобного хранения всех нужных данных, нужно будет использовать Базу Данных. Но для этого сначала предстояло знакомство с языком программирования MySql и Sql и соответствующими средами разработки: MySqlWorkbench и MySqlNotifier. По рекомендациям родственника одного из исполнителей, в основном вся работа с БД происходила в MySqlWorkbench, вторая среда разработки использовалась лишь для конкретных случаев, где всего функционала первой не хватало. Эти среды разработки создавали максимально удобное и простое взаимодействие с БД, а именно их создание, верстка в различные разрешения, и полноценную работу с конкретной БД. Так же была использована возможность поднятия собственного сервера для полноценной проверки функциональности собственной БД и отладки. После работы с базой данных нужно было придумать способ ее заполнения. Вручную это заняло бы несколько дней, поэтому был выбран самый простой способ в этой ситуации – с помощью парсера данных проходить по страницам ТП и заполнять собственную БД для дальнейших работ с ней. Было опробовано множество бесплатных парсеров, но у каждого из них был свой недостаток. С помощью простого перебора был найден оптимальный – ParseHub. Его преимущество были в понятной и безошибочной работе и интуитивно понятной и простой настройке, а так же в его функционал входила сразу нужная конвертация в .json (Java Script Object Notation) файл, что сильно облегчило работу с базой данных, так как теперь к ней можно было обращаться через JavaScript, а именно на JavaScript написан сайт. Потом появился вопрос с реализацией самого процесса работы бота. Нужно было добиться со стороны бота полноценной возможности автоматически производить покупку нужного покупателю предмета на ТП Steam. Первая проблема, которая была обнаружена, это полная неясность процесса работы ботов и способы их программирования. Поэтому перед тем, как была начата работа по созданию кода собственного бота, была проделана огромная работа для ознакомления с миром программирования ботов и их реализации. Были изучены, как и простейшие боты для соцсетей, так и многозадачные боты для различных игр. Вся эта работа была проделана для основного понимания этой индустрии. Так после изучения сайта ТП были найдены элементы, значение которых и отвечало за возможность покупать вещи на ней. Соединив это открытие и знания, полученные раньше, было решено создавать бота на основе языка JavaScript. Так же знакомый отца одного из исполнителей проекта подсказал некие фишки и дал полезные советы. Само написание не было очень трудным, потому что ранее человеку, который выполнял эту часть проекта, уже предоставлялась возможность ознакомления с синтаксисом JavaScript. Из-за этого написание кода не составило значительных проблем. Проблемы оказались только в отладке и полноценном тесте бота. Для более удобной демонстрации этого кода, весь код был закоменчен. После этого другой участник команды, начал конкретное изучение JavaScript-a. Был просмотрен массивный объём видеороликов по синтаксису JavaScript. Так же были просмотрены видеоролики о создании WEB-приложений на JavaScript. Были изучены Node.JS, JSON, AXIOS и AJAX в рамках проекта.//Часть Пончика

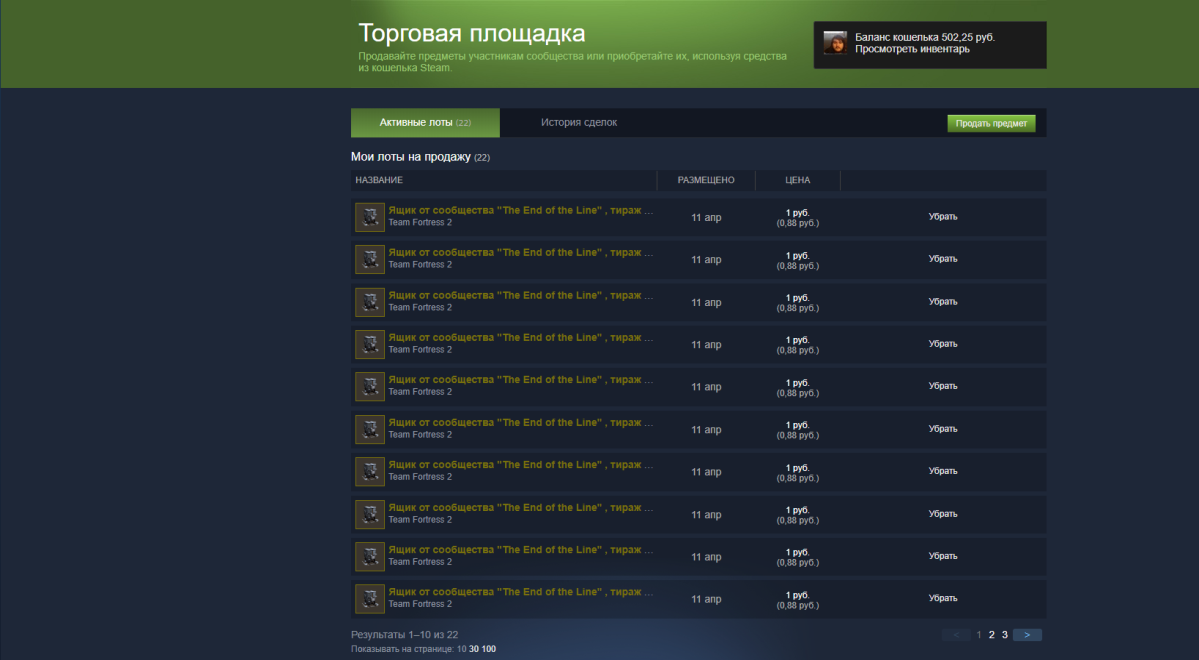
Анализ предметной Области

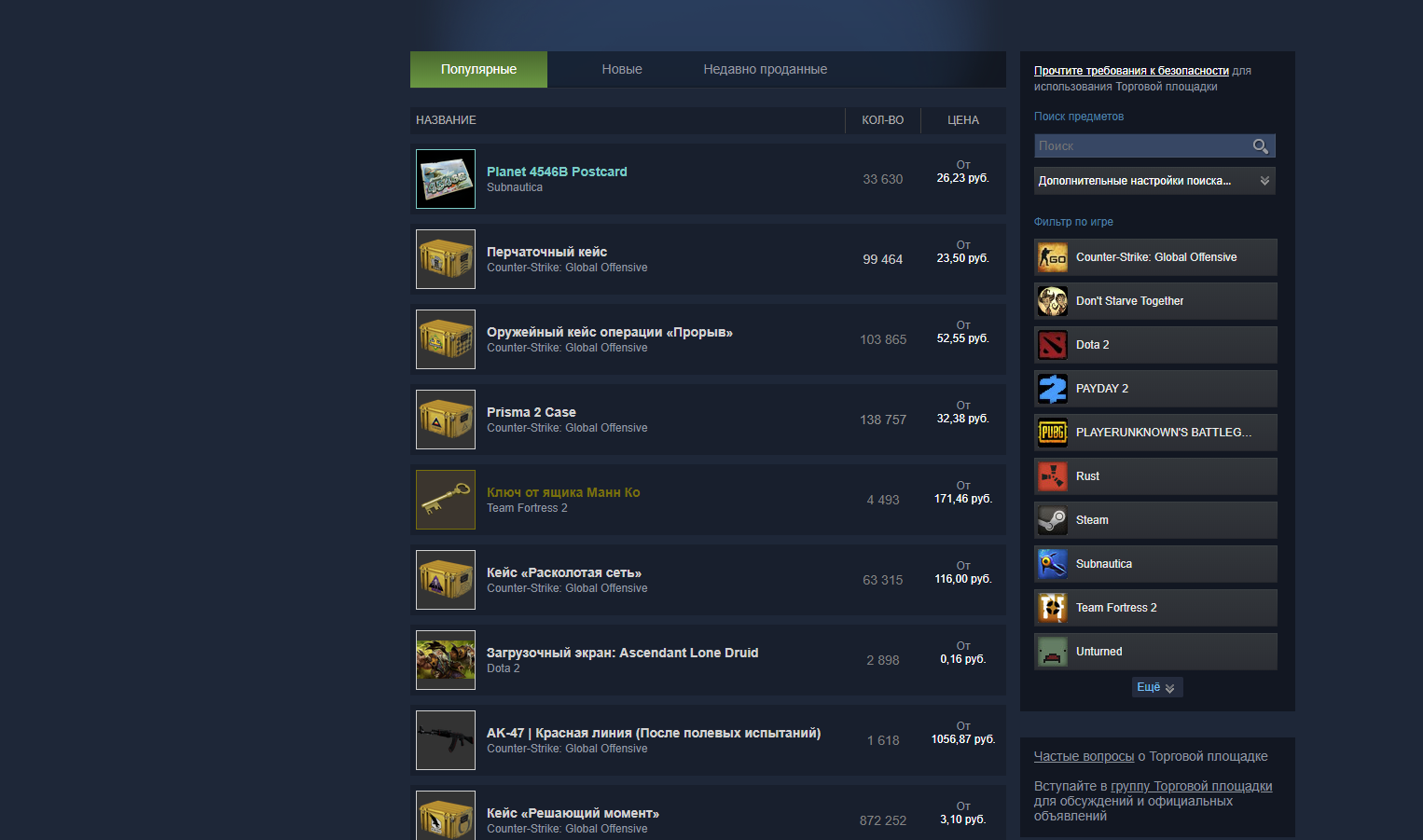
A **Steam**  — [онлайн-сервис](https://ru.wikipedia.org/wiki/%D0%9E%D0%BD%D0%BB%D0%B0%D0%B9%D0%BD-%D1%81%D0%B5%D1%80%D0%B2%D0%B8%D1%81" \o "Онлайн-сервис) [цифрового распространения](https://ru.wikipedia.org/wiki/%D0%A6%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%B0%D1%8F_%D0%B4%D0%B8%D1%81%D1%82%D1%80%D0%B8%D0%B1%D1%83%D1%86%D0%B8%D1%8F) [компьютерных игр](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%B8%D0%B3%D1%80%D0%B0) и программ, разработанный и поддерживаемый компанией [Valve](https://ru.wikipedia.org/wiki/Valve" \o "Valve). Steam выполняет роль средства [технической защиты авторских прав](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D1%85%D0%BD%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B5_%D1%81%D1%80%D0%B5%D0%B4%D1%81%D1%82%D0%B2%D0%B0_%D0%B7%D0%B0%D1%89%D0%B8%D1%82%D1%8B_%D0%B0%D0%B2%D1%82%D0%BE%D1%80%D1%81%D0%BA%D0%B8%D1%85_%D0%BF%D1%80%D0%B0%D0%B2), платформы для [многопользовательских игр](https://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B3%D0%BE%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D0%BA%D0%B0%D1%8F_%D0%B8%D0%B3%D1%80%D0%B0) и [потокового вещания](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%B2%D0%BE%D0%B5_%D0%BC%D1%83%D0%BB%D1%8C%D1%82%D0%B8%D0%BC%D0%B5%D0%B4%D0%B8%D0%B0), а также [социальной сети](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D1%86%D0%B8%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C) для игроков. Программный клиент Steam также обеспечивает установку и регулярное обновление игр, [облачные сохранения игр](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D0%BB%D0%B0%D1%87%D0%BD%D0%BE%D0%B5_%D1%85%D1%80%D0%B0%D0%BD%D0%B8%D0%BB%D0%B8%D1%89%D0%B5_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85), текстовую и голосовую связь между игроками.По состоянию на осень 2015 года через Steam распространяется порядка 10 тысяч игр для операционной системы [Microsoft Windows](https://ru.wikipedia.org/wiki/Microsoft_Windows" \o "Microsoft Windows), свыше 2300 игр для [macOS](https://ru.wikipedia.org/wiki/MacOS" \o "MacOS) и свыше 1500 игр для [Linux](https://ru.wikipedia.org/wiki/Linux" \o "Linux), на которые действуют ежедневные и «срединедельные» скидки, а также регулярные распродажи. Количество активных учётных записей Steam превышает 125 миллионов. В январе 2018 года количество одновременно находящихся онлайн пользователей Steam достигло 18 миллионов.



**Торговая площадка сообщества Steam** — это под-секция, которая позволяет игрокам покупать внутриигровые предметы за валюту Steam-кошелька.

Предметы могут быть выставлены на продажу по цене за которую должен будет заплатить покупатель. Из стоимости всех предметов в продаже вычитается комиссия, составляющая 10%. Покупателю следует обратить внимание на то, что ещё существует комиссия на перечисление денег на баланс Steam-кошелька, составляющая 5%. Чтобы продавать предметы на Торговой площадке сообщества Steam, пользователь должен купить хотя бы одну игру в Steam, после этого проходит 30 дней и только тогда вы сможете что-то купить или продать.





**Node или Node.js** — программная платформа, основанная на движке V8 (транслирующем JavaScript в машинный код), превращающая JavaScript из узкоспециализированного языка в язык общего назначения. Node.js добавляет возможность JavaScript взаимодействовать с устройствами ввода-вывода через свой API (написанный на C++), подключать другие внешние библиотеки, написанные на разных языках, обеспечивая вызовы к ним из JavaScript-кода. Node.js применяется преимущественно на сервере, выполняя роль веб-сервера, но есть возможность разрабатывать на Node.js и десктопные оконные приложения (при помощи NW.js, AppJS или Electron для Linux, Windows и macOS) и даже программировать микроконтроллеры (например, tessel, low.js и espruino). В основе Node.js лежит событийно-ориентированное и асинхронное (или реактивное) программирование с неблокирующим вводом/выводом.

**Робот, или бот,  а также интернет-бот, www-бот и тому подобное** — специальная [программа](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0), выполняющая [автоматически](https://ru.wikipedia.org/wiki/%D0%90%D0%B2%D1%82%D0%BE%D0%BC%D0%B0%D1%82%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F) и/или по заданному [расписанию](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D1%81%D0%BF%D0%B8%D1%81%D0%B0%D0%BD%D0%B8%D0%B5) какие-либо действия через [интерфейсы](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81), предназначенные для [людей](https://ru.wikipedia.org/wiki/%D0%9B%D1%8E%D0%B4%D0%B8).

При обсуждении компьютерных программ слово употребляется в основном в применении к [Интернету](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82). В наше время есть боты, которые используются в большинстве сфер. Боты могут автоматически отвечать на телефонный звонок, противостоять в шахматном поединке, даже обучать вас химии. В наше время все чаще боты заменяют людей, сохраняя людское время. Так же боты используются в корыстных целях, например в фишинге коммерческих почт. Так же сейчас растет количество ботов, наделенных искусственным интеллектом.

**База данных (БД)** — это организованная структура, предназначенная для хранения, изменения и обработки взаимосвязанной информации, преимущественно больших объемов. [Базы данных](https://hostiq.ua/wiki/migrate-mysql-database/) активно используются для динамических сайтов со значительными объемами данных — часто это интернет-магазины, порталы, корпоративные сайты. Такие сайты обычно разработаны с помощью серверного языка программирования (как пример, PHP) или на основе CMS (как пример, WordPress), и не имеют готовых страничек с данными по аналогии с HTML-сайтами. Странички динамических сайтов формируются «на лету» в результате взаимодействия скриптов и баз данных после соответствующего запроса клиента к веб-серверу.

## Axios -- Библиотека [Axios](https://github.com/axios/axios), предназначенная для выполнения HTTP-запросов, основана на промисах. Она подходит для использования в среде Node.js и в браузерных приложениях. Библиотека поддерживает все современные браузеры, и, в том числе, IE8+.

**Парсинг сайтов или парсинг контента** — это процесс извлечения данных любого сайта в сети Интернет.

Типичным примером парсинга контента является копирование списка контактов из некоего веб-каталога. Однако извлечение и сохранение данных с веб-страницы в таблицу Excel работает только с небольшими объемами данных и занимает значительное время. Чтобы обработать крупные массивы данных, нужна автоматизация. И здесь в дело вступают веб-парсеры.

Парсинг сайтов осуществляется при помощи специальной программы "веб-парсера" или "бота" или "веб-паука" (обычно все эти понятия используются как синонимы). Веб-парсер сканирует веб-страницы, загружает контент, извлекает из него нужные данные и затем сохраняет их в файлах или базе данных.

**AJAX** это Асинхронный JavaScript и XML. Это набор методов веб-разработки, которые позволяют веб-приложениям работать асинхронно — обрабатывать любые запросы к серверу в фоновом режиме. Подождите, что снова AJAX? Давайте рассмотрим каждый термин отдельно и разберём, что такое AJAX. JavaScript — это хорошо известный язык кодирования. Помимо прочего, он управляет динамическим контентом веб-сайта и позволяет динамически взаимодействовать с пользователем. XML — это ещё один вариант языка разметки, такой как HTML, как следует из названия — расширяемый язык разметки. Если HTML предназначен для отображения данных, XML предназначен для хранения и переноса данных. И JavaScript, и XML работают асинхронно в AJAX. В результате любое веб-приложение, использующее AJAX, может отправлять и извлекать данные с сервера без необходимости перезагрузки всей страницы.

**Автобай** – это функция на Тогровой Площадке Steam, с помощью которой, вы сможете покупать нужные вам предметы без вашего собственного участия. Вы заранее находите интересующий вас предмет, и выставляете цену, за который вы его купите. Как только любой пользователь выставит его по нужной вам цене, сама система Steam, скупит его автоматически.

# B алгоритм Кнута-Морриса-Пратта (КМП)

[Алгоритм Кнута-Морриса-Пратта](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%9A%D0%BD%D1%83%D1%82%D0%B0_%E2%80%94_%D0%9C%D0%BE%D1%80%D1%80%D0%B8%D1%81%D0%B0_%E2%80%94_%D0%9F%D1%80%D0%B0%D1%82%D1%82%D0%B0) используется для поиска подстроки (образца) в строке. Кажется, что может быть проще: двигаемся по строке и сравниваем последовательно символы с образцом. Не совпало, перемещаем начало сравнения на один шаг и снова сравниваем. И так до тех пор, пока не найдем образец или не достигнем конца строки. Функция примерно такая:

// простой поиск образца в строке

// возвращает индекс начала образца в строке или -1, если не найден

int find(char \*образец, char \*строка)

{

// i-с какого места строки ищем

// j-с какого места образца ищем

for (int i=0;строка[i];++i) {

for (int j=0;;++j) {

if (!образец[j]) return i; // образец найден

if(строка[i+j]!=образец[j]) break;

}

// пока не найден, продолжим внешний цикл

}

// образца нет

return -1;

}

Простой случай поиска'игла' — образец'стогистогстогигстогстогиглстогстогигластогигластог' — строка поискаСложный случай поискаaabbaabaabaabbaaabaabaabaabaabbaabbФункция работает и довольно шустро, если образцы и строка «хорошие». Хорошие — это когда внутренний цикл прерывается быстро (на 1-3 шаге, скажем, как в простом случае). Но если образец и строка содержат часто повторяющиеся вложенные куски (как сложном случае выше), то внутренний цикл обрывается ближе к концу образца и время поиска оценивается как О(<длина образца>\*<длина строки>). Если длина строки 100тыс, а длина образца 100, то получаем О(10млн). Конечно, реально редко встретишь образец длиной 100, но в олимпиадных задачах «на поиск» это обычное дело, поэтому там простой поиск часто не подходит.А если строка — это длинный текст, а образец-фрагмент слова, и надо найти все вхождения этого фрагмента, причем на лету, по мере набора слова (замечали, как быстро это делают браузеры)? Алгоритм КМП находит все вхождения образца в строку и его скорость О(<длина образца>+<длина строки>), поэтому на больших текстах/образцах или на слабых процессорах (как в низкобюджетных сотовых) он вне конкуренции.А теперь посмотрим на заголовок? Почему «маленькое»? Потому, что изюминка КМП — это префикс-функция, а она действительно маленькая. А почему «чудо»? Потому что, он вроде как решает совсем другую задачу и это решение, после некоторого чудесного трюка, превращается в решение задачи поиска всех вхождений образца в строку.Для того, чтобы понять, что и как делает префиксная функция, посмотрим на сложную строку

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| a | a | b | a | a | b | a | a | a | a | b | a | a | b | а | a | a | b |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
| 0 | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 2 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3 |

Строка под ней — номер (позиция) символа в строке (для удобства описания алгоритма считаем номер с 1), а самая нижняя строка- массив M длин префиксов, ключ к пониманию префикс-функции.Возьмем символ с номером 7 (это a) и для K от 1 до 6 рассмотрим строки-префиксы (подстрока, начинающаяся с первого индекса строки) и суффиксы (подстрока, последний символ которой в строке в позиции 7 ( это «наш» символ a) длины K.

|  |  |  |
| --- | --- | --- |
| K | префикс | суффикс |
| 1 | а | a | тут все просто: префикс — это первый символ строки, а суффикс-наш символ a |
| 2 | aa | ba |  |
| 3 | aab | aba |  |
| 4 | aaba | aaba | самое длинное совпадение, причем здесь и ниже суффикс и префикс начали перекрываться (как фрагменты строки поиска) |
| 5 | aabaa | baaba |  |
| 6 | aabaab | abaaba |  |

Обратите внимание: для длины 4 суффикс (как последовательность символов) совпадает с префиксом и это максимальное значение K при котором суффикс совпадает с префиксом. Именно оно и заносится в соответствующую позицию (7) массива длин префиксов. Можно заметить, что для K=7 префикс тоже совпадет с суффиксом, поскольку это одна и та же строка. Но этот тривиальный случай нам не подходит, т.к. для работы алгоритма нужны именно подстроки.Обозначим S-исходная строка, S(n) -начало (префикс) строки S длины n, S[n]-символ в позиции n строки S. M[n] значение в массиве, S(M[n])- та самая строка, которая являемся префиксом и суффиксом максимальной длины для позиции n (для краткости обозначим её P(n)). Строка P(n) как бы «виртуальная», она не формируется и никуда не пишется. Это просто начальный фрагмент1 исходной строки S длины M[n]. И этот начальный фрагмент1 совпадает (как последовательность символов) с фрагментом2 длины M[n], последний символ которого в позиции n. Если M[n]=0, то совпаденией нет.Имеем: позиция 7 массива заполнена значением М[7]=4, самая длинная строка P(7)='aaba' длины 4 (естественно), надо перейти к позиции 8 и заполнить M[8]. Можно тупо посчитать все префиксы и суффиксы длиной от 1 до 7, сравнить их и записать максимальную длину в позицию 8. Но мы пойдем другим путем (вслед за КМП). Пусть найдена максимально длинная строка P(8) длины k, которая префикс и суффикс для позиции 8. Строка p7 из первых k-1 символов является префиксом и суффиксом для позиции k-1. Не факт, что для 7й позиции она самая длинная. Однако, если оказалось, что p7=P7, то P8 — это расширение P7 на один символ. Чтобы проверить, можно ли расширить P7 на одну позицию, надо проверить, совпадает ли добавляемый в суффикс символ (это символ S[8]=a) со следующим символом префикса. Следующий символ префикса a находится в позиции М[7]+1=5 (подумайте, почему так). Если совпал (а в нашем случае он совпал), то задача выполнена — М[8]=М[7]+1, а P(8)=P(7)+символ в 8 позиции S[8]=a. Получаем P(8)='aabaa'. При успешном расширении надо всего одно сравнение для вычисления очередного значения массива. Кстати, отсюда следует, что при движении вдоль массива значения могут возрастать максимум на 1.Для удобства повторю сложную строку, чтобы не нужно было перемещаться вверх-вниз.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| a | a | b | a | a | b | a | a | a | a | b | a | a | b | а | a | a | b |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
| 0 | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 2 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3 |

Теперь другой случай — P8 расширить не удалось, т.е. символ S[9]=a не совпал с символом строки S в позиции M[8]+1=6 b. Суффикс расширяется легко (поскольку новый символ просто дописывается в конец), а с префиксом проблемы, т.к. добавляемый в суффикс символ может не совпасть с очередным символом префикса. Если префикс P(k) не подходит, надо найти другой такой префикс, покороче, у которого такой же суффикс и попробовать расширить его. Но префикс покороче, причем с таким же суффиксом — это S[M[M[k]]), т.к. при заполнении массива М каждый элемент содержит длину максимально длинного префикса с таким же суффиксом. Поэтому, если не удалось расширить S(M[k]) пробуем так же расширить S(M[M[k]]) и т.д, пока не совпадет или пока не получим нулевую длину (тогда очередной символ S надо просто сравнить с 1м символом строки S). Цикл перебора походящих префиксов заканчивается очень быстро, потому, что вся нужная для этого информация уже сидит в массиве М.Для нашей строки P(8) — это просто расширение P(7) на один символ, понадобилось 1 сравнение. Однако P(8) не удалось расширить до P(9), поскольку S[9]=a, а S[M[8]+1=6] =b. Раз не подошел префикс P8 длины M[8]=5, пробуем префикс длины M[5]=2. Он тоже не подходит: S[2+1] =b. Пробуем префикс длины M[2]=1 и его можно расширить, т.к. S[1+1] =a. Поэтому M[9]=2, на единицу больше длины расширяемого префикса. Для заполнение M[10] надо 2 сравнения (можете проверить). А вот чтобы заполнить элементы с 11 по 17, понадобится по одному сравнению. В результате расчет значений массива занимает время порядка О(длина строки).Итак, с алгоритмом заполнения более-менее ясно.

## Переходим к трюку.

Для нахождения образца ааbаа в строке ааbааbааааbааbаааb склеим образец со строкой вот так ааbаа@ааbааbааааbааbаааb и вызовем для нее префикс-функцию для заполнения массива.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| a | a | b | a | a | @ | a | a | b | a | a | b | a | a | a | a | b | a | a | b | а | a | a | b |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 |
| 0 | 1 | 0 | 1 | 2 | 0 | 1 | 2 | 3 | 4 | 5 | 3 | 4 | 5 | 2 | 2 | 3 | 4 | 5 | 3 | 4 | 5 | 2 | 3 |

Символ '@' играет роль разделителя, его заведомо нет ни в образце, ни в строке поиска (нужно подобрать именно такой символ). Посмотрим на позиции 11, 14, 19, 22 массива. Значения в массиве равны 5, это означает, что суффикс длиной 5 (фрагмент строки поиска) совпадает с 5 символами префикса. А 5 символов префикса — это и есть образец для поиска! Алгоритм поиска получатся такой — склеиваем с помощью разделителя образец и строку поиска, передаем «склейку» префиксной функции и потом ищем в массиве элементы, равные длине образца, Можно заметить, что значений больше длины образца не будет из-за символа-разделителя, а значения, равные длине образца могут появиться только в позициях, соотвествующих исходной строке поиска. Склеенная строка имеет длину <длина образца>+<длина строки>, поэтому время расчета оценивается как О(<длина образца>+<длина строки>), как и говорилось в начале статьи. Объем необходимого префикс-функции буфера равен <длина образца>+<длина строки>, но можно модифицировать префикс-функцию так, чтобы хватило буфера <длина образца> (пример модификации в дополнении)

## Префикс-функция

А теперь примеры префикс-функции. Отличие от описания выше в том, что, как принято в Си-языках, индексы считаются с 0.

#### Пример на C++

Функция возвращает вектор длин префиксов строк, а строка передается как string (не надо вычислять длину)

vector<size\_t> prefix\_function (string s)

{

size\_t n = s.length();

vector<size\_t> pi(n); // в i-м элементе (его индекс i-1) количество совпавших символов в начале и конце для подстроки длины i.

// p[0]=0 всегда, p[1]=1, если начинается с двух одинаковых

for (size\_t i=1; i<n; ++i)

{

// ищем, какой префикс-суффикс можно расширить

size\_t j = pi[i-1]; // длина предыдущего префикса-суффикса, возможно нулевая

while ((j > 0) && (s[i] != s[j])) // этот нельзя расширить,

j = pi[j-1]; // берем длину меньшего префикса-суффикса

if (s[i] == s[j])

++j; // расширяем найденный (возможно пустой) префикс-суффикс

pi[i] = j;

}

return pi;

}

## 

Результат

На данный момент можно смело заявить, что этот проект готов на 75%. Было изучено большое количество новых материалов, начиная с простейшего языка Web-программирования, продолжая его сложными методами, для углубленной работы с ним, а заканчивая различными тонкостями работы по созданию ботов. Первый испольнитель изучил HTML, CSS, JavaScript, Ajax, NodeJS, ExpressJS, Axios, Json, PHP. Это все необходимо было для красивого оформления сайта, для удобного взаимодействия пользователя с ботом. Так же все это ему понадобилось для написания задней части кода, которая отвечает за связь между кодами, и отклик программы с пользователем. Так же он изучил ранее ему не известный алгоритм Кнута-Морриса-Пратта(КМП). Второй исполнитель изучил новые материалы вдругом направление. Изучил БД и способы работы с ней. Так же изучил работы с парсером и заполнением json файлев. И самое главное, изучал способы работы с ботами и управление ими. Проект закончен на 75 процентов, так как был протестирован лишь на небольшой БД, с небольшим количеством наименований. Так же возможность передачи пользователю его купленной вещи не до конца оформлена. В остальном, проект удалось реализовать во всех направлениях, не отходя от первоначальных задумок. Так же дизайну сайта отвелось не слишком много времени, потому что концентрация была направлена на создание полностью функционального проекта, в первую очередь. Выполнить весь проект на 100 процентов не удалось возможным, так как материала, который необходимо было изучить, оказалось в разы больше, чем изначально задумывалось.

Вывод

Перспективы дальнейшей разработки весьма четкие и ясные. В первую очередь это выполнение проекта на 100 % от первоначальной задумки(Авторизация через Steam аккаунт). Так же тщательная проработка дизайна, для наиболее понятной коммуникации сайта с пользователем. Надо полностью разобраться с возможностью бота стабильно покупать и получать свои купленные предметы на ТП, не только знакомым исполнителей, но и любому пользователю ТП. Так же возможность увеличивать количество игр, из которых будет возможно совершать автозакупку с помощью разработанного бота. Возможно\* добавление 3D изображений покупаемого предмета, для полного ознакомления с ним. Актуальность этой темы будет всегда, пока существует Steam и его пользователи. Так же из-за пандемии актуальность увеличилась, чем изначально предполагалась. Спрос рождает предложения. Этому утверждению можно самостоятельно убедиться, заглянув на растущий онлайн Steam пользователей.