**Автономная некоммерческая
общеобразовательная организация "Физтех-лицей"
(АНОО «Физтех-лицей» им. П.Л. Капицы)**

XX научно-практическая конференция

«Старт в инновации»

Telegram бот помощник для учеников

Выполнили:
Машталер Максим 8В
Вольнов Иван 8Б
Николаенко Виталий 8В
Руководитель:
В.В.Мерзляков

Московская область, г. Долгопрудный

2021 г.

ВВЕДЕНИЕ

Пожалуй, наша работа началась с того, что мы стали замечать, что с электронным журналом часто возникают проблемы: пропадает домашнее задание, перестает корректно работать приложение, не отображаются некоторые сообщения, что обусловливает **актуальность** проведенной нами работы.
Изначально у нас была одна задача — собирать все домашнее задание на неделю, чтобы экономить время на ручном просмотре электронного журнала каждый день. Про телеграм бота мы тогда и не думали, просто хотелось вместо ежедневного просмотра электронного журнала получать домашнее задание на неделю одним нажатием кнопки. К тому же, задание в электронном журнале периодически обновляется и добавляется, так что, чтобы ничего не упустить, и нужен автоматический сборщик.

Хотелось за прогон скрипта получать домашнее задание на всю неделю. API сайта в открытом доступе нет, поэтому было принято решение пропарсить HTML-страницы, по тегам выдергивая нужное. Выглядело адекватно, так что дело быстро дошло до первой реализации.

В итоге мы решили сделать телеграм бота, который будет парсить информацию о предстоящих контрольных работах, домашнем задание, расписание, новых сообщениях с сайта элжура каждый день и хранить в своей памяти.

Мы выдвинули **гипотезу**: телеграм бот существенно поможет лицеистам.

Поставили **цель**: написать Telegram бота помощника лицеиста

Для достижения поставленной цели были сформулированы **задачи**:

1. Выбрать более удобный язык программирования для написания бота.
2. Изучить библиотеку “**Selenium**” языка программирования Python.
3. Изучить HTML код сайта элжура.
4. Написать парсера элжура.
5. Изучить структуру Telegram ботов.
6. Написать Telegram бота и подключить к нему парсера.
7. Оформить бота и запустить его на сервере.

Затем мы приступили к выполнению работы.

Мы изучили различную литературу по данной теме; провели эксперименты по сравнению разных способов создания парсеров, с целью выявить плюсы и минусы разных библиотек; создали парсер и протестировали его, сравнили, на каком языке программирования лучше писать телеграм бота, написали бота и подключили к нему

нашего парсера. На основе проделанной нами работы мы сделали выводы, подтверждающие выдвинутую нами гипотезу.

В своей работе мы использовали следующие **методы исследования:**

- Анализ материалов, взятых из Интернета и печатных источников.
- Экспериментальные исследования различных языков программирования и способов парсинга информации с сайтов
- Написание кода на языке программирования Python с использованием библиотеки Selenium

ОСНОВНАЯ ЧАСТЬ

Работа над проектом

Первая реализация. Минимальный функционал

К реализации мы подошли с опытом работы в языке программирования Python и знаниями html, web-архитектуры сайтов. Для первых набросков нам понадобились модули requests и Beautiful Soup 4. Какую информацию и откуда выдергивать можно понять по структуре сайта. Первым делом — собираем адреса домашних заданий, которые есть в электронном журнале. Из html-страницы нам надо считать домашние задания, чтобы потом пройтись по ним и вывести пользователю. Примерно так происходит сборка списка дз.

Выбор языка программирования

Для написание нашей работы мы выбирали между несколькими языками: Python, C#, PHP, JavaScript. В итоге мы выбрали Python как самый удобный язык для создания ботов, ведь в нем есть все необходимые библиотеки для выполнения нужной нами задачи.

Python — высокоуровневый язык программирования общего назначения с динамической типизацией и автоматическим управлением памятью, ориентированный на повышение производительности разработчика, читаемости кода и его качества, а также на обеспечение переносимости написанных на нём программ. Язык является полностью объектно-ориентированным .Особенностью языка является выделение блоков кода пробельными отступами.

Изучение ботов

Бот — специальная программа, выполняющая автоматически или по заданному расписанию какие-либо действия через интерфейсы, предназначенные для людей.

Обычно боты предназначаются для выполнения работы, однообразной и повторяемой, с максимально возможной скоростью.

Боты находят также применение в условиях, когда требуется лучшая реакция по сравнению с возможностями человека (например, игровые боты, боты для интернет-аукционов и т.д) иногда для имитации действий человека (например, боты для чатов).

Чат-бот может выдать достаточно адекватный ответ на вопрос, сформулированный на правильном русском языке (или любом другом, работа с которым поддерживается). Такие боты часто применяются для сообщения прогноза погоды, результатов спортивных соревнований, курсов валют, биржевых котировок и тому подобное.

Почему мы выбрали платформу Telegram для создания бота?

1. Telegram полностью конфиденциален.
2. Для ботов выделено ограниченное место на серверах Telegram.
3. Боты не могут сами начать общение с пользователем.
4. Создание ботов в Telegram полностью бесплатно.
5. Боты Telegram никогда не останавливают свою работу если они подключены к серверу.

Изучение Telegram ботов.

Боты — специальные аккаунты в Telegram, созданные для того, чтобы автоматически обрабатывать и отправлять сообщения. Пользователи могут взаимодействовать с ботами при помощи сообщений, отправляемых через обычные или групповые чаты. Логика бота контролируется при помощи HTTPS запросов к нашему API для ботов.

У ботов Telegram есть много уникальных возможностей — например, кастомизированные клавиатуры, дополнительные интерфейсы для команд по умолчанию, внешнее связывание и специальные режимы приватности для групп.

Создание бота
BotFather — главный бот, который управляет всей инфраструктурой ботов Telegram. При помощи него меняются настройки у существующих ботов и создаются новые. Чтобы писать функции бота, нам надо получить токен в BotFather. После того как мы получили токен можно приступать к написанию функций бота. Для написания функций бота мы выбрали язык программирования - Python.

Написание функций

- Для начала нам нужно импортировать библиотеку и подключить токен бота:

```
import telebot;
bot = telebot.TeleBot('%токен%');
```

- Теперь объявим метод для получения текстовых сообщений:

```
@bot.message_handler(content_types=['text'])
def get_text_messages(message):
```

  В этом участке кода мы объявили на какой тип сообщений будет отвечать бот и обрабатывать сообщение .

- Теперь добавим в наш метод немного функционала: если пользователь напишет нам команду «/help», то скажем пользователю написать:

  «Расписание уроков»

  «Домашнее задание»

  «Новые сообщения»

```
if message.text == "Расписание уроков":
    bot.send_message(message.from_user.id, " ")
elif message.text == "Домашнее задание":
    bot.send_message(message.from_user.id, " ")
else:
    bot.send_message(message.from_user.id, " ")
```

- Чтобы бот все время проверял пришли ему сообщения или нет надо написать строку:

```
bot.polling(none_stop=True, interval=0)
```

- Чтобы сделать нашего бота более удобным для пользователя мы написали код для кнопок:

```
keyboard = types.InlineKeyboardMarkup(); #наша клавиатура
key_yes = types. InlineKeyboardButton(text='Расписание уроков', callback_data='Расписание уроков');
keyboard.add(key_yes); #добавляем кнопку в клавиатуру
key_no= types.InlineKeyboardButton(text='Домашнее задание', callback_data='Домашнее задание');
keyboard.add(key_yes); #добавляем кнопку в клавиатуру
key_no= types.InlineKeyboardButton(text='Новые сообщения', callback_data='Новые сообщения');
```

Дальше мы начали думать, как связать нашего парсера и бота, и перебрали несколько вариантов: добавить парсера напрямую в код бота или сохранять данные из элжура с помощью парсера в отдельный файл и после бот будет брать оттуда данные и отправлять пользователю либо использовать базы данных.

Объединение парсера и бота

Чтобы объединить возможности парсера и бота, мы использовали библиотеку SQLitestudio

SQLite – **это** библиотека, написанная на языке C, реализующая SQL механизм работы с данными, другими словами, движок баз данных. (Рис. 1)
Движок **SQLite** не является отдельно работающим процессом, с которым взаимодействует программа, а представляет собой библиотеку, с которой программа компонуется, и движок становится составной частью программы. Такой подход уменьшает накладные расходы, время отклика и упрощает программу.
SQLite хранит всю базу данных (включая определения, таблицы, индексы и данные) в единственном стандартном файле на том компьютере, на котором исполняется программа.
Несколько процессов или потоков могут одновременно без каких-либо проблем читать данные из одной базы.
Запись в базу можно осуществить только в том случае, если никаких других запросов в данный момент не обслуживается; в противном случае попытка записи оканчивается неудачей, и в программу возвращается код ошибки. Другим вариантом развития событий является автоматическое повторение попыток записи в течение заданного интервала времени.
SQLite поддерживает динамическое типизирование данных.

Что такое парсер?

Парсер - часть программы, отвечающая за изучение контента в авторежиме и поиск нужных фрагментов. В ходе синтаксического анализа исходный текст преобразуется в структуру данных, обычно — в дерево, которое отражает синтаксическую структуру входной последовательности и хорошо подходит для дальнейшей обработки. Как правило, результатом синтаксического анализа является синтаксическое строение предложения, представленное либо в виде дерева зависимостей, либо в виде дерева составляющих, либо в виде некоторого сочетания первого и второго способов представления. Его используют для обработки и сбора данных, преобразовывая их в дальнейшем в структурированный формат. Обычно парсер предпочитают использовать для работы с текстами.

Программа позволяет сканировать наполнение веб-страниц, различные результаты выдачи поисковых систем, текст, картинки и множество сведений. С её помощью можно идентифицировать большой объем непрерывно обновляемых значений.

Что делает парсер?

Алгоритм в соответствии с программой сверяет конкретный набор символов с тем, что нашлось в интернете. Стоит отметить, что программное обеспечение может иметь разные форматы представления, стилистику оформления, варианты доступности, языки и многое другое. Работа всегда происходит в несколько этапов. Сначала происходит поиск сведений, загрузка и скачивание. Далее значения извлекаются из кода веб-страницы так, что материал отделяется от программного кода страницы. В итоге формируется отчет в соответствии с заданными требованиями напрямую в базу данных или сохраняется в текстовой файл. Парсер сайта дает много преимуществ при работе с массивами данных. Например, высокую скорость обработки материалов и их анализ даже в огромном объеме. Также автоматизируется процесс отбора сведений.

Для чего нужен парсер?

Это и всевозможное извлечение контактных сведений при разработке базы потенциальных клиентов, так и поиск непосредственно по ней в собственном веб-ресурсе. При этом будут найдены не внешние ссылки, а вхождение поискового запроса, вбитый пользователем. Необходимость в парсере возникает, например, при сборе ссылок SEO специалистами. Все они знают, что такое язык поисковых запросов и как отражается это в их работе. Они используют парсер для того, чтобы оценить количество ссылок и ссылаемых ресурсов. Когда требуется работать с большим количеством ссылок, парсер – незаменимый инструмент в оптимизации. Он без проблем соберет информацию и распарсит ее в удобном виде.

Мы использовали парсер в своей работе для сбора данных из элжура про домашние задания, расписания, предстоящие контрольные.

Написание парсера

Selenium WebDriver — это инструмент для автоматизации действий веб-браузера. В большинстве случаев используется для тестирования Web-приложений, но этим не ограничивается. В частности, он может быть использован для решения рутинных задач

администрирования сайта или регулярного получения данных из различных источников (сайтов).

В июне 2004 года разработчик Jason Huggins написал на языке JavaScript библиотеку, названную «JavaScriptTestRunner» позже переименованную в «Selenium Core» и предназначенную для запуска тестов в браузере. Название «Selenium» стало использоваться после того, как в одном из своих электронных писем (email) Huggins пошутил о конкурирующем проекте, имеющем название «Mercury Interactive QuickTest Professional».

Selenium WebDriver — это в первую очередь набор библиотек для различных языков программирования. Эти библиотеки используются для отправки HTTP запросов драйверу (отсюда и название WebDriver), с помощью протокола JsonWireProtocol, в которых указано действие, которое должен совершить браузер в рамках текущей сессии. Примерами таких команд могут быть команды нахождения элементов по локатору, переход по ссылкам, парсинг текста страницы/элемента, нажатие кнопок или переход по ссылкам на странице веб-сайта. Проектом Selenium и сообществом поддерживается работа с браузерами Microsoft Internet Explorer, Google Chrome, Mozilla Suite и Mozilla Firefox под управлением операционных систем Microsoft Windows, Linux и Apple Macintosh.

В рамках проекта Selenium выпускается инструмент «Selenium IDE» — расширение к браузеру Firefox, представляющее собой библиотеку Selenium с графическим интерфейсом (GUI). Расширение позволяет записывать, сохранять и воспроизводить сценарии тестирования web-страниц. Сценарии сохраняются в формате HTML в виде таблицы.

Команды **Selenium**

Элемент Locators помогают Selenium идентифицировать элемент HTML, на который ссылается команда. Все эти локаторы можно идентифицировать с помощью плагина.

Действия - это команды, которые управляют состоянием приложения. После выполнения, если действие не выполняется, выполнение текущего теста прекращается.

Accessors оценивают состояние приложения и сохраняют результаты в переменной, которая используется в утверждениях.

Assertions - Утверждения позволяют нам проверять состояние приложения и сравнивать с ожидаемым. Он используется в трех режимах, а именно: - утверждать (assert), проверять (verify) и ждать (waitfor).

Вывод

В итоге мы создали telegram бота, к которому подключили парсера ЭЛЖУРа написанный на языке Python, который поможет Вам когда электронный журнал будет работать некорректно, вы сможете узнать информацию о расписании, домашнем задании и ближайших контрольных за одно нажатие кнопки.

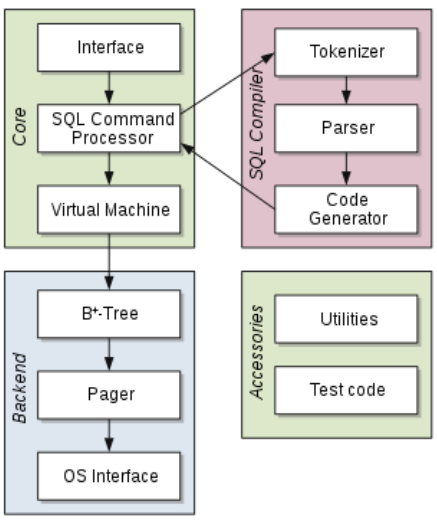

Рис. 1 Архитектура SQLite

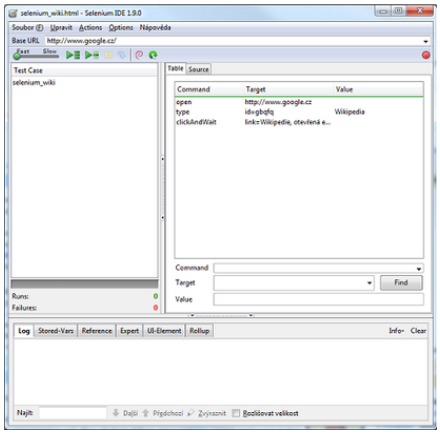

Рис. 2 **Selenium**

Браузер	Операционная система	Разработчик
Chromium/Google Chrome	Windows/macOS/Linux	Google
Firefox	Windows/macOS/Linux	Mozilla
Edge	Windows 10	Microsoft
Internet Explorer	Windows	Selenium Project
Safari	macOS El Capitan и более новые	Apple
Opera	Windows/macOS/Linux	Opera

Рис. 3 Поддерживаемые Selenium платформы

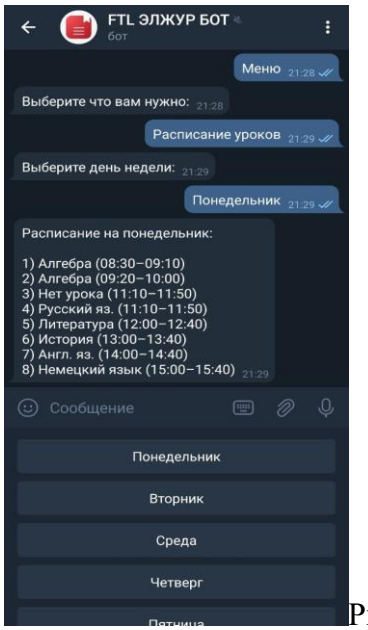

Рис. 4 Наш telegram bot