



ТЗ Автоматизация сайт быстрокабель

1. Цель и результат

Цель:

Автоматически раз в неделю собирать данные о заявках и предложениях с сайта

<https://bystrokabel.ru/offers> (с авторизацией по логину/паролю),

и сохранять результат в один Excel-файл

Результат:

Готовый скрипт/сервис + инструкция по запуску через внешний оркестратор (n8n / Яндекс / Just AI), который:

- Авторизуется на сайте по логину и паролю.
- Проматывает страницу до конца, чтобы подгрузить все заявки.
- По каждой заявке собирает данные заявки и всех предложений во вкладке «Все предложения».
- Раз в неделю (по расписанию) добавляет в единый Excel-файл новый “снимок” новых/изменённых заявок и их предложений.

2. Область работ

Входит:

- Парсинг (чтение) данных с сайта bystrokabel.ru/offers с авторизацией.
- Обработка динамической подгрузки (скролл вниз до конца списка).
- Сохранение данных в Excel и CSV по заданной структуре (2 листа).

- Базовый механизм идентификации новых/обновлённых заявок.
- Логирование работы и ошибок.
- Уведомления об ошибках (Telegram).
- Инструкция по интеграции с оркестратором.

Не входит:

- Любые действия на сайте (создание заявок, отправка предложений).
 - Любая аналитика/рекомендации по ценам — только подготовка данных.
 - Разработка BI-отчётов / дашбордов (только примеры сводных таблиц в самом Excel).
-

3. Функциональные требования

3.1. Авторизация

- Скрипт должен осуществлять авторизацию на `bystrokabel.ru` с использованием логина и пароля.
- Логин и пароль не хардкодятся в коде, а читаются из **переменных окружения**:
 - `BYSTROKABEL_LOGIN`
 - `BYSTROKABEL_PASSWORD`
- Если авторизация не удалась:
 - Записать ошибку в лог (с указанием причины, если доступно).
 - Отправить уведомление в Telegram.
 - Не создавать/не изменять файлы данных.

3.2. Загрузка всех заявок (скролл)

- Скрипт должен проматывать страницу `https://bystrokabel.ru/offers` вниз до тех пор, пока:
 - Не перестанут появляться новые заявки (признак — количество карточек/строк не растёт).
- Допускается использование любого подхода (Selenium, Playwright, Puppeteer, headless-браузер, эмуляция API вызовов), главное — гарантированная загрузка всех заявок.

3.3. Собираемые данные по заявке

Для каждой заявки на странице нужно собрать и сохранить:

- `capture_datetime` — дата и время съёма данных (по времени сервера).
- `request_number` — номер заявки (например, `122098`).
- `request_deadline_raw` — строка с датой “до ... включительно” (как на сайте).
- `delivery_city` — город из фразы `Поставка в <город> ...` .
- `supply_type` — тип поставки: `"из наличия"` или `"под заказ"` (как на сайте).
- `cable_name` — наименование кабеля (например, `ВВГнг(А)-LS 1x240 - 1кВ (мн)`).
- `cable_length_raw` — длина кабеля, как на сайте (например, `0,450 км`).
- `request_raw_text` — полный сырой текст блока заявки (для надёжности).

Все поля должны сохраняться **без изменения формата**, кроме явных служебных (`capture_datetime`).

3.4. Собираемые данные по предложениям (“Все предложения”)

Для каждой заявки необходимо:

- Перейти/раскрыть вкладку «**Все предложения**».
- По каждому предложению собрать:
 - `capture_datetime` — дата и время съёма (та же, что у заявки).
 - `request_number` — номер заявки (для связи с листом “Заявки”).
 - `offer_index` — порядковый номер предложения для данной заявки (1, 2, 3...).
 - `offer_city_and_info_raw` — строка вида `г. Москва` , `г. Москва (Собственное производство...)` и т.п. — полностью, как на сайте.
 - `delivery_days_raw` — срок доставки, как на сайте (например, `10 дн.` , `2 дн.`).
 - `price_with_unit_raw` — цена с единицей измерения (например, `7 360 000,00 руб./км с НДС` , `784,00 руб./м с НДС`).
 - `price_change_raw` — изменение цены в процентах или спец-значение как на сайте:
 - `+0,4%`
 - `-1,2%`
 - `min` и т.п.
 - `stock_type_raw` — наличие/под заказ/под производство и подобные пометки (например, `наличие` , `под производство`).

- `offer_raw_text` — полный текст предложения.

Важно:

Никакой нормализации цен (к руб./км и т.п.) подрядчик не делает. Всё хранится “как есть”.

4. Логика “только новые/обновлённые” заявки

Требование: при еженедельном запуске сохранять **только новые или изменённые** заявки относительно предыдущих запусков.

Реализовать так:

1. Подрядчик ведёт служебное хранилище “последнее известное состояние заявок”

(это может быть:

- отдельный лист в том же Excel-файле, или
- отдельный служебный файл

2. Ключ заявки: `request_number` .

3. При каждом запуске:

- Загружаются все заявки.
- Для каждой заявки сравниваются основные поля с последним сохранённым состоянием:
 - `request_deadline_raw`
 - `delivery_city`
 - `supply_type`
 - `cable_name`
 - `cable_length_raw`
- Если номер заявки новый (нет в хранилище) **или** хотя бы одно из полей изменилось:
 - заявка считается “новой/обновлённой” и её текущие данные + все её предложения записываются в рабочие листы Excel как новый снимок (с текущим `capture_datetime`);
 - новое состояние заявки сохраняется в служебное хранилище.
- Если всё совпадает — заявка считается “без изменений” и **не записывается** в рабочие листы этого запуска.

4. Таким образом:

- Исторический трекинг цен и условий по заявке ведётся по изменениям.

- В рабочей истории можно видеть, когда какая заявка изменилась и как поменялись предложения.
-

5. Формат хранения: Excel и CSV

5.1. Структура файлов

- Один общий файл Excel, например:

`bystrokabel_market_data.xlsx`

- Директория хранения настраиваемая (через config/переменную окружения), по умолчанию —

`./data/`.

5.2. Структура Excel

Лист 1: **requests** (Заявки)

Обязательные колонки:

- `capture_datetime`
- `request_number`
- `request_deadline_raw`
- `delivery_city`
- `supply_type`
- `cable_name`
- `cable_length_raw`
- `request_raw_text`

Лист 2: **offers** (Предложения)

Обязательные колонки:

- `capture_datetime`
- `request_number`
- `offer_index`
- `offer_city_and_info_raw`
- `delivery_days_raw`

- `price_with_unit_raw`
- `price_change_raw`
- `stock_type_raw`
- `offer_raw_text`

Требования:

- При каждом запуске новые строки **добавляются** в конец листов.
 - Существующие строки не изменяются (история не переписывается).
 - файлы должны быть в кодировке UTF-8; разделитель — `;` или `,` (важно задокументировать в инструкции).
-

6. Расписание и запуск

- Частота: **1 раз в неделю, по четвергам**.
- Реализация расписания — на стороне внешнего оркестратора (n8n / Яндекс / Just AI).

Требования к скрипту:

- Предоставить точку запуска:
 - либо CLI-команду, например:
`python main.py run`
 - либо HTTP-endpoint (по согласованию), если оркестратор удобнее работает по HTTP.
 - Выходной код процесса:
 - `0` — всё успешно;
 - не `0` — ошибка (чтобы оркестратор видел сбой).
-

7. Технические требования

7.1. Стек

- Основной язык реализации: **Python**.
- Разрешается использовать JS (Node.js) для фронтowego/браузерного парсинга, если это ускоряет работу:

- в этом случае нужно чётко описать, что запускается из оркестратора.
- Рекомендуемые библиотеки (необязательно, но логично):
 - для браузера: Playwright;
 - для Excel: `pandas` , `openpyxl` , стандартные средства Python.

7.2. Docker (опционально, но желательно)

- Желательно предоставить `Dockerfile` для развёртывания:
 - образ, который:
 - устанавливает все зависимости,
 - запускает скрипт по `CMD` / `ENTRYPOINT` (например, `python main.py`).
 - Внутри контейнера логин/пароль и прочие настройки подаются через **переменные окружения**.
-

8. Логирование и уведомления

8.1. Логирование

- Писать лог в файл, например: `logs/bystrokabel_parser.log` .
- Уровни логов:
 - `INFO` — основные этапы (старт, авторизация, количество заявок, успешное завершение).
 - `ERROR` — ошибки (авторизация, недоступность сайта, изменение структуры HTML и т.п.).
- Лог минимум должен содержать:
 - дату/время,
 - уровень,
 - краткое описание.

8.2. Уведомления (Telegram)

- Реализовать отправку уведомлений в Telegram при **критических ошибках**:
 - не удалось авторизоваться;
 - не удалось загрузить страницу/получить список заявок;
 - необработанное исключение.

- Настройки через переменные окружения, пример:

- `TG_BOT_TOKEN`
- `TG_CHAT_ID`

- Формат сообщения: кратко, по делу:

- что за ошибка,
 - на каком этапе,
 - дата/время по серверу.
-

9. Настройки и конфигурация

Все чувствительные и средозависимые параметры — через переменные окружения или отдельный config-файл:

Обязательные:

- `BYSTROKABEL_LOGIN`
- `BYSTROKABEL_PASSWORD`

Желательные:

- `DATA_DIR` — путь к директории для данных (`./data` по умолчанию).
 - `LOG_DIR` — путь к логам (`./logs` по умолчанию).
 - `TG_BOT_TOKEN` , `TG_CHAT_ID` — для уведомлений.
-

10. Документация для подрядчика

Подрядчик должен предоставить:

1. **README** (1–2 страницы), где описано:

- как установить зависимости;
- как запускать скрипт вручную;
- какие переменные окружения нужны;
- как интегрировать в p8n / другой оркестратор (пример сценария: “запуск раз в неделю по четвергам”).

2. **Пример Excel-файла** с тестовыми данными:

- два листа: `requests` и `offers` ;
- несколько строк, полностью собранных с реального сайта (но без реального логина/пароля в репозитории).

3. Примеры сводных таблиц внутри Excel (по самому файлу):

- пример сводной таблицы по:
 - средним ценам по `cable_name` и `delivery_city` ,
 - количеству заявок по типу `supply_type` и городу.
- Достаточно 1–2 примера, чтобы показать, что структура данных пригодна для аналитики.

11. Критерии приёмки

Работа считается принятой, если выполняются условия:

1. Скрипт успешно:

- авторизуется на сайте;
- загружает все заявки (проверка вручную по счёту карточек);
- для каждой заявки собирает все предложения во вкладке “Все предложения”.

2. В Excel-файле:

- создаются листы `requests` и `offers` с указанными колонками;
- данные в минимум 10 произвольно выбранных заявках и их предложениях полностью совпадают с сайтом (ручная проверка).

3. При втором запуске:

- в файле появляются новые строки только по тем заявкам, где были изменения или появились новые;
- старые строки не перезаписываются и не удаляются.

4. При имитации ошибки (неверный пароль/выключенный интернет):

- создаётся запись в логе с уровнем `ERROR` ;
- приходит уведомление в Telegram;
- рабочие данные (Excel) не портятся.

