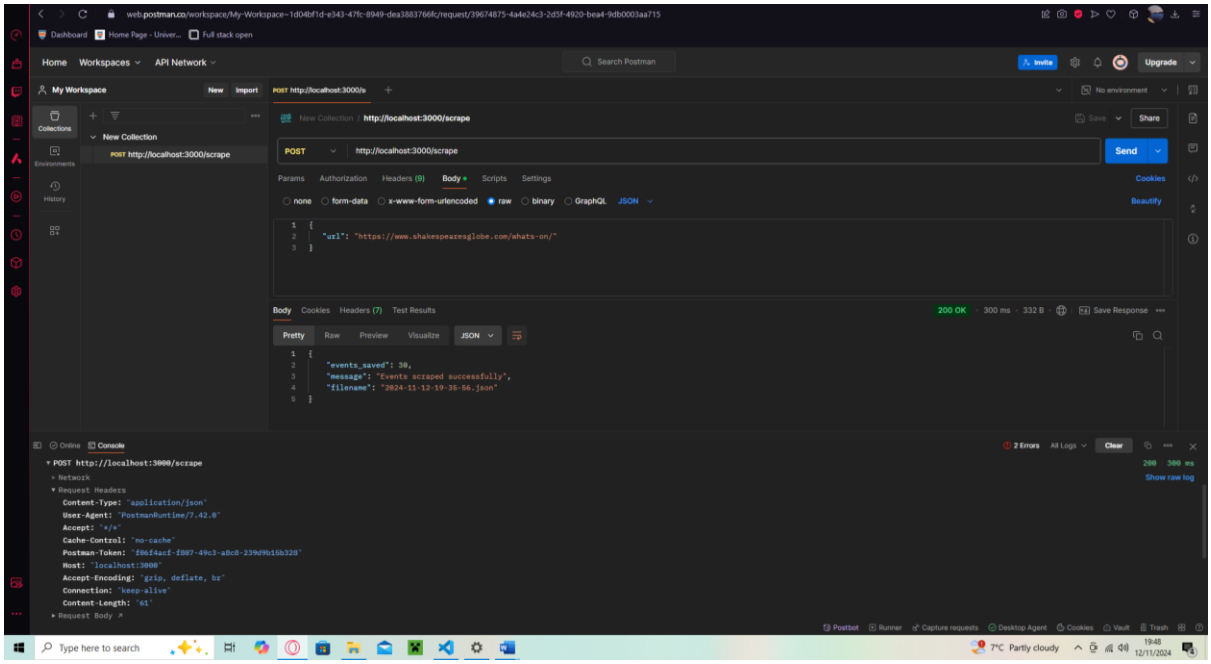
JavaScript web scraper is the preferred choice due to its advantages working on dynamic sites over python and the familiarity of the language compared to python.

https://github.com/BBarrettGreening/web-scraper

Example scraper designed to scrape the name, dates and locations of shows from the Shakespeare Globe theatre website.

Postman initially used to send POST requests to the scrape endpoint with the website URL given in the request body



The application successfully scraped all 30 productions on the website and saved them in a JSON file when running on the localhost via node.js – Need list of attributes Stagely wants from all sites.

Currently could be possible to do multiple sites by making the fields it is parsing from dynamic and allow any url in the request then the request body would be formatted like this.

- Would be very time consuming to define a layout for each site but possible

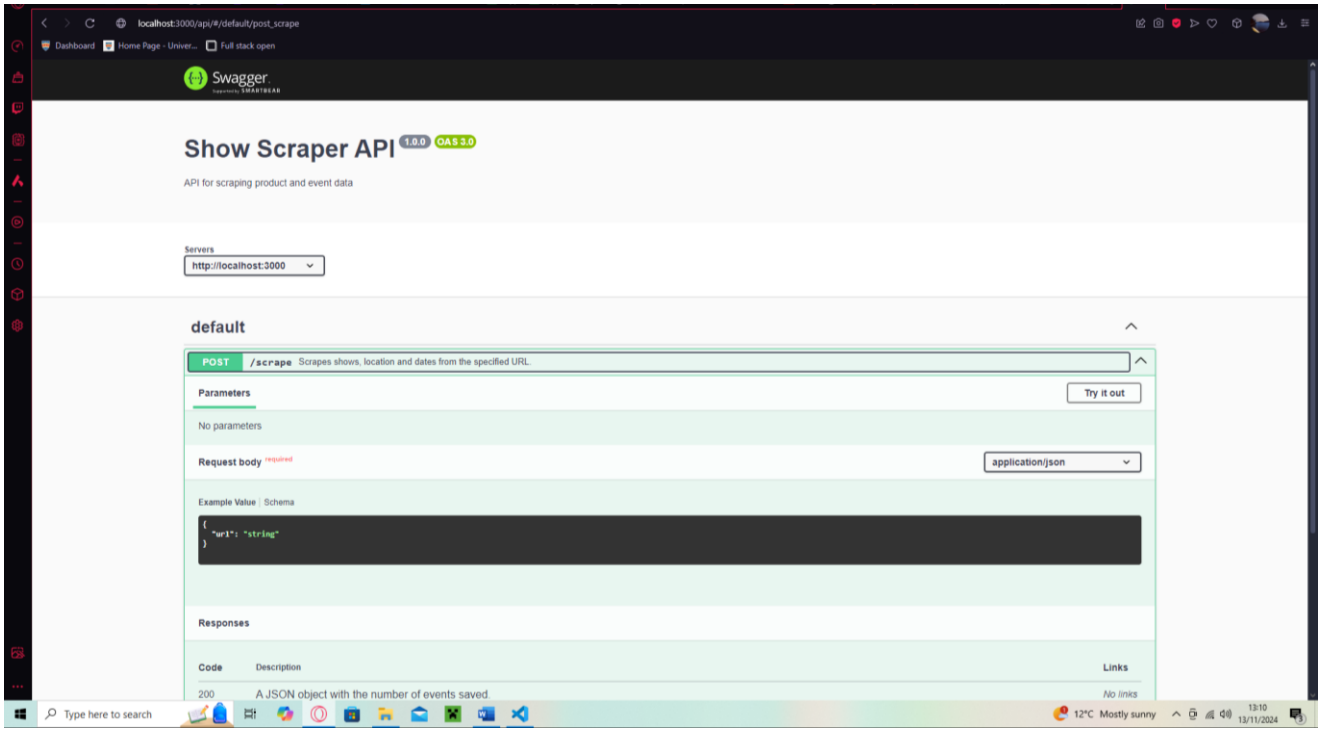
```
{
  "sites": [
    {
      "url": "https://www.shakespearesglobe.com/some-page",
      "selectors": {
        "title": ".c-event-card__title",
        "date": ".c-event-card__date",
        "location": ".c-event-card__venue",
        "link": ".c-event-card__cover-link"
      }
    },
    {
      "url": "https://www.anotherwebsite.com/events",
      "selectors": {
        "title": ".event-title",
        "date": ".event-date",
        "location": ".event-location",
        "link": ".event-link"
      }
    }
  ]
}
```

Ideally another service will scan the site HTML and return the attributes we're searching for to be passed to the scraper body. (Dynamically handle site layout changes)

The saved JSON outputs can be handled by another service which would sort the shows into a database (SQL database to sort in Stagely's preferred format)

The database sorted by shows could then be accessed and pushed to the content management system by another service (Another API which takes information from the database and autofill sections for the user)

To prevent having to use a third-party website to access the API endpoints swagger can also be implemented which will generate a UI for the web scraper actions.



```
// Swagger setup
const swaggerUi = require('swagger-ui-express');
const swaggerJSDoc = require('swagger-jsdoc');

const swaggerOptions = {
  swaggerDefinition: {
    openapi: '3.0.0',
    info: {
      title: 'Show Scraper API',
      version: '1.0.0',
      description: 'API for scraping product and event data',
    },
    servers: [
      {
        url: `http://localhost:${port}`,
      },
    ],
  },
  apis: ['./src/server.js', './src/scraper.js'],
};
```

```
/**
 * @swagger
 * /scrape:
 *   post:
 *     summary: Scrapes shows, location and dates from the specified URL.
 *     requestBody:
 *       required: true
 *       content:
 *         application/json:
 *           schema:
 *             type: object
 *             properties:
 *               url:
 *                 type: string
 *                 description: The URL to scrape.
 *     responses:
 *       200:
 *         description: A JSON object with the number of events saved.
 *         content:
 *           application/json:
 *             schema:
 *               type: object
 *               properties:
 *                 events_saved:
 *                   type: integer
 *                   description: The number of events scraped and saved.
 *                 message:
 *                   type: string
 *                   description: Success message.
 *                 filename:
 *                   type: string
 *                   description: Name of the file where events are saved.
 *       400:
 *         description: Invalid URL provided.
 *       500:
 *         description: Error scraping events.
 */
```