Krok 1: Przygotowanie Środowiska

1. \*\*Zainstaluj Node.js\*\*

- Pobierz i zainstaluj Node.js z [oficjalnej strony](https://nodejs.org/).

2. \*\*Utwórz nowy projekt\*\*

- Otwórz terminal i utwórz nowy folder dla projektu:

```sh

mkdir KRS-Data-Collector

cd KRS-Data-Collector

```

3. \*\*Zainicjuj projekt Node.js\*\*

- Zainicjuj projekt, tworząc plik `package.json`:

```sh

npm init -y

```

### Krok 2: Instalacja Potrzebnych Bibliotek

Zainstaluj wymagane biblioteki:

```sh

npm install axios exceljs yaml

```

### Krok 3: Struktura Projektu

Utwórz następującą strukturę folderów i plików:

```

KRS-Data-Collector/

│

├── config/

│ └── config.yaml # Plik konfiguracyjny

├── data/

│ └── companies.xlsx # Plik Excel z danymi firm

├── scripts/

│ ├── fetchData.js # Skrypt do pobierania danych

│ ├── processData.js # Skrypt do przetwarzania danych

│ └── saveToExcel.js # Skrypt do zapisywania danych do Excela

├── logs/

│ └── app.log # Plik logów

└── main.js # Główny skrypt uruchamiający

```

### Krok 4: Plik konfiguracyjny

\*\*config/config.yaml\*\*

```yaml

apiUrl: 'https://api.krs.example.com'

apiKey: 'YOUR\_API\_KEY'

```

### Krok 5: Skrypty

\*\*scripts/fetchData.js\*\*

```javascript

const axios = require('axios');

const yaml = require('yaml');

const fs = require('fs');

const config = yaml.parse(fs.readFileSync('config/config.yaml', 'utf8'));

async function fetchData() {

try {

const response = await axios.get(config.apiUrl, {

headers: { 'Authorization': `Bearer ${config.apiKey}` }

});

return response.data;

} catch (error) {

console.error('Error fetching data:', error);

}

}

module.exports = fetchData;

if (require.main === module) {

fetchData().then(data => console.log(data));

}

```

\*\*scripts/processData.js\*\*

```javascript

function processData(rawData) {

return rawData.map(entry => ({

company\_name: entry.name,

creation\_date: entry.creation\_date,

activity: entry.activity,

}));

}

module.exports = processData;

if (require.main === module) {

const fetchData = require('./fetchData');

fetchData().then(rawData => {

const data = processData(rawData);

console.log(data);

});

}

```

\*\*scripts/saveToExcel.js\*\*

```javascript

const ExcelJS = require('exceljs');

const path = require('path');

async function saveToExcel(data, filepath = 'data/companies.xlsx') {

const workbook = new ExcelJS.Workbook();

const worksheet = workbook.addWorksheet('Companies');

worksheet.columns = [

{ header: 'Company Name', key: 'company\_name', width: 30 },

{ header: 'Creation Date', key: 'creation\_date', width: 20 },

{ header: 'Activity', key: 'activity', width: 30 }

];

worksheet.addRows(data);

worksheet.autoFilter = 'A1:C1';

await workbook.xlsx.writeFile(filepath);

console.log('Data saved to Excel file successfully');

}

module.exports = saveToExcel;

if (require.main === module) {

const fetchData = require('./fetchData');

const processData = require('./processData');

fetchData().then(rawData => {

const data = processData(rawData);

saveToExcel(data);

});

}

```

\*\*main.js\*\*

```javascript

const fetchData = require('./scripts/fetchData');

const processData = require('./scripts/processData');

const saveToExcel = require('./scripts/saveToExcel');

async function main() {

const rawData = await fetchData();

const data = processData(rawData);

await saveToExcel(data);

}

main();

```

### Krok 6: Uruchomienie Skryptu

Uruchom skrypt w terminalu:

```sh

node main.js

```

### Krok 7: Automatyzacja

#### Linux (cron)

1. Otwórz terminal i edytuj crontab:

```sh

crontab -e

```

2. Dodaj linijkę, aby uruchamiać skrypt co tydzień:

```sh

0 0 \* \* 0 /usr/bin/node /path/to/KRS-Data-Collector/main.js

```

#### Windows (Task Scheduler)

1. Otwórz Task Scheduler.

2. Utwórz nową zaplanowaną operację.

3. Skonfiguruj, aby uruchamiała się co tydzień.

4. Jako akcję wybierz uruchomienie programu Node.js z argumentem ścieżki do `main.js`.

### Podsumowanie

Postępując zgodnie z powyższymi krokami, powinieneś mieć działający skrypt, który co tydzień pobiera dane z API KRS-u, przetwarza je i zapisuje do pliku Excel. Możesz dostosować te kroki do swoich specyficznych potrzeb, np. dodając logowanie błędów, bardziej zaawansowane przetwarzanie danych, czy inne funkcje.