

به نام خدا



گزارش فنی پروژه

## Costumized Web Crawler

اعضای گروه:

نگار باروتی، مریم نصر، علیرضا موسوی خونساری، صالح جوانمرد پای قلعه

استاد پروژه:

دکتر ترک لادانی

منبع کد:

<https://github.com/Maryam-Nsi/Web-crawler>

تاریخ: ۱۴۰۱/۱۱/۶

|         |                           |
|---------|---------------------------|
| ۳.....  | معرفی اعضای گروه          |
| ۳.....  | چکیده                     |
| ۳.....  | تعریف پروژه               |
| ۳.....  | احساس نیاز                |
| ۴.....  | کاربرد کنونی              |
| ۴.....  | بدنه                      |
| ۴.....  | نیازمندی‌ها               |
| ۴.....  | ابزارها                   |
| ۵.....  | اسناد                     |
| ۵.....  | مدل کسب و کار             |
| ۶.....  | گانت چارت                 |
| ۷.....  | مستند API                 |
| ۱۰..... | موجودی پایگاه داده        |
| ۱۱..... | Prototype                 |
| ۱۱..... | ساختار دنباله‌های پردازشی |
| ۱۲..... | بررسی نتایج               |
| ۱۴..... | سناریو تست                |
| ۱۴..... | پیشنهادهات                |
| ۱۵..... | نتیجه گیری                |
| ۱۵..... | منابع                     |

## معرفی اعضای گروه

- نگار باروتی: توسعه‌دهنده frontend و پیاده سازی فرم ها
- مریم نصر: توسعه‌دهنده frontend و طراح ui
- علیرضا موسوی خونساری: توسعه‌دهنده frontend و backend و پیاده‌سازی نمایش اطلاعات و قسمت url injection
- صالح جوانمرد: توسعه‌دهنده backend و پیاده‌سازی data extraction و html fetcher

## چکیده

مستند تهیه شده حاوی اطلاعاتی است که شما را با جزئیات فنی و پیاده سازی این پروژه آشنا می‌کند. این پروژه با استفاده از توسعه دو فاز جدا گانه frontend و backend توسعه یافته است و در قسمت deploy از سروری با سیستم عامل ubuntu کمک گرفته شده است. در این مستند ابتدا با تعریف پروژه آشنا می‌شوید، سپس با نیازمندی‌ها، مراحل توسعه و اسناد ارائه پروژه مواجه می‌شوید. با توجه به تعریف محدود پروژه در این فاز، این پروژه قابل گسترش است. در قسمت پیشنهادات با این جزئیات بیشتر آشنا خواهید شد. در نهایت منابعی که با خواندن آنها آشنایی عمیق‌تری با پروژه حاصل می‌شود آورده شده است.

این پروژه با عنوان web crawler تهیه شده است. وظیفه این پروژه یافتن اطلاعات با استفاده از مختصات ورودی کاربر است. خروجی این پروژه شامل جدولی از داده هاست که با ویژگی هایی که کاربر تایین کرده تشکیل می‌شود. خروجی این پروژه می‌تواند به عنوان داده تمرین برای ساختن مدل استفاده شود.

## تعریف پروژه

### احساس نیاز

سایت ترپ را در نظر بگیرید. وظیفه این سایت واکنشی اطلاعاتی مانند قیمت و نام کالا از صفحه‌های مختلف اینترنت است. در این فرایند یک موازنه‌ای میان دقت و حجم اطلاعات خروجی برقرار است. اگر در حالتی دقت این اطلاعات را کاهش و حجم اطلاعات خروجی را افزایش دهیم کاربر سایت ترپ با اطلاعات زیادی مواجه می‌شوند که بعضا اشتباه هستند. این عملکرد که برای کاربر سایت ترپ، نکته منفی تلقی می‌شود می‌تواند با تغییر تعریف مشتری ارزش افزوده تولید کند. با اعمال عملکرد جدید و افزایش تولید داده و کاهش دقت می‌توان دیتاست هایی با ابعاد ده‌ها میلیون برای استفاده دانشمندان علم داده تولید کرد.

الگوریتم های داده کاوی بعضا دارای قدمتی بیشتر از پردازنده‌های امروزی دارند. با ورود این پردازنده‌ها این الگوریتم ها وارد حوزه کاربرد شدند. پس از این مرحله چالش علم داده کاوی خود داده است. ارزش داده در دنیای امروز خیلی زیاد است. امروزه شاهد این هستیک که شرکت‌های بزرگ داده های خود را چه به صورت خام و چه به صورت ساختارمند به فروش می‌رسانند. حال اگر ابزاری توسعه یابد که داده را به صورت ساختاریافته نه تنها از یک شرکت بلکه از کل اینترنت واکنشی کند، می‌تواند ارزش افزوده زیادی تولید کند.

## کاربرد کنونی

در حال حاضر این پروژه می تواند داده را از sitemap های دارای فرمت xml استخراج کند. با قرار دادن فایل در منابع پروژه می توانید صفحه های اینترنتی درون فایل را خزش کنید. عملکرد این پروژه با ورودی دیگری به نام selector کامل می شود. به صورتی که در یک صفحه اینترنتی و با انتخاب یک scc selector می توانید این داده را از تمام sitemap واکشی کنید.

## بدنه

### نیازمندی ها

۱. این سیستم باید با استفاده از عنوان دامنه یک سایت فایل sitemap.xml را خروجی دهد.
۲. این سیستم باید از یک فایل sitemap.xml لیستی از آدرس های صفحات اینترنتی را خروجی دهد.
۳. این سیستم باید html هر صفحه را دریافت کند.
۴. این سیستم باید هر صفحه اینترنتی را با استفاده از scc selector پارس کند.
۵. این سیستم باید نتیجه هر پارس را به علاوه آی دی هر درخواست خزش ذخیره و دریافت کند.
۶. این سیستم باید برای هر درخواست خزش یک آی دی منحصر به فرد در نظر بگیرد و اطلاعات هر خزش تنها مربوط به همان باشد.
۷. این سیستم باید برای هر درخواست خزش حداکثر صفحات مورد واکاوی را تایین کند.
۸. این سیستم باید با استفاده از آی دی خزش نتایج خزش را برگرداند.

### ابزارها

- Ant Design
- Tailwind
- Axios
- React
- Maven
- Spring Boot
- Apache Camel
- Spring Doc
- H2 database
- Lombok
- Seda Broker
- Jsoup

اسناد

مدل کسب و کار

## مدل کسب و کار




Figure 1



rFigure

- سمت backend پروژه با استفاده دو API پشتیبانی می‌شود. در ادامه جزئیات بیشتری ارائه خواهد شد.



Explore

This tool can integrate your data from hole network. 0.0.1 OAS3

/api-doc

---

**Servers**

---

**task-controller** ^

GET /v1/ You can get your task result with its id and pagination. v

POST /v1/ You can start your data set creation task based on your configuration input. v

---

**Schemas** ^

FeatureDTO >

TaskConfigurationInputDTO >

TaskIdOutputDTO >

RecordsOutputDTO >

swagger ui rFigure

- شروع کار پروژه با ارسال یک **config** آغاز می‌شود. این کانفیگ دارای یک لیست از **feature** ها است که دارای یک نام و یک **scc selector** یا عنوان **reach result** هستند. این اطلاعات برای استخراج اطلاعات از هر صفحه از دامنه ورودی به کار گرفته خواهد شد. در ضمن این **API** به صورت **async** طراحی شده است و با در دست داشتن یک کد پیگیری خروجی می‌توان نتایج خزش را دریافت کرد.

POST /v1/ You can start your data set creation task based on your configuration input.

Parameters
Try it out

No parameters

Request body required
application/json

Example Value | Schema

```

{
  "hostname": "string",
  "uriStartRegex": "string",
  "datasetSize": 0,
  "webPageVisitSize": 0,
  "requestPerSecond": 0,
  "features": [
    {
      "name": "string",
      "richesultPath": "string",
      "selector": "string"
    }
  ]
}

```

Responses

| Code    | Description  | Links    |
|---------|--|----------|
| default | It contains a task id. You can follow your task with that. | No links |

Media type
\*/

Controls Accept header.

Example Value | Schema

```

{
  "id": "3fa85f64-5717-4562-b3fc-2c963f66afa6"
}

```

Figure ۴ درخواست خزش



- در این قسمت با استفاده از taskId و اطلاعات pagination نتایج را دریافت می کنید. خروجی شامل یک لیست از record ها است که هر رکورد شامل اطلاعاتی است که با selector های ورودی استخراج شده است. هر record نظیر یک صفحه اینترنتی خزش شده است.

GET /v1/ You can get your task result with its id and pagination.

Parameters
Try it out

| Name   | Description |
|--|-------------|
| taskId * required<br>string<br>(query)               | taskId      |
| pageNumber * required<br>integer(\$int32)<br>(query) | pageNumber  |
| pageSize * required<br>integer(\$int32)<br>(query)   | pageSize    |

Responses

| Code    | Description  | Links    |
|---------|--|----------|
| default | <p>It contains list of records in json type.</p> <p>Media type<br/>*/*</p> <p>Controls Accept header.</p> <p>Example Value   Schema</p> <pre>[   {     "additionalProp1": "string",     "additionalProp2": "string",     "additionalProp3": "string"   } ]</pre> | No links |

دریافت نتایج خزش مورد نظر Figure 5



Figure 6 شمای dto ها

موجودی پایگاه داده  
این پروژه تنها دارای یک موجودی است.

| record  |         |
|---------|---------|
| id      | int     |
| task_id | varchar |
| data    | varchar |

Figure 7

Figure 8

ساختار دنباله‌های پردازشی

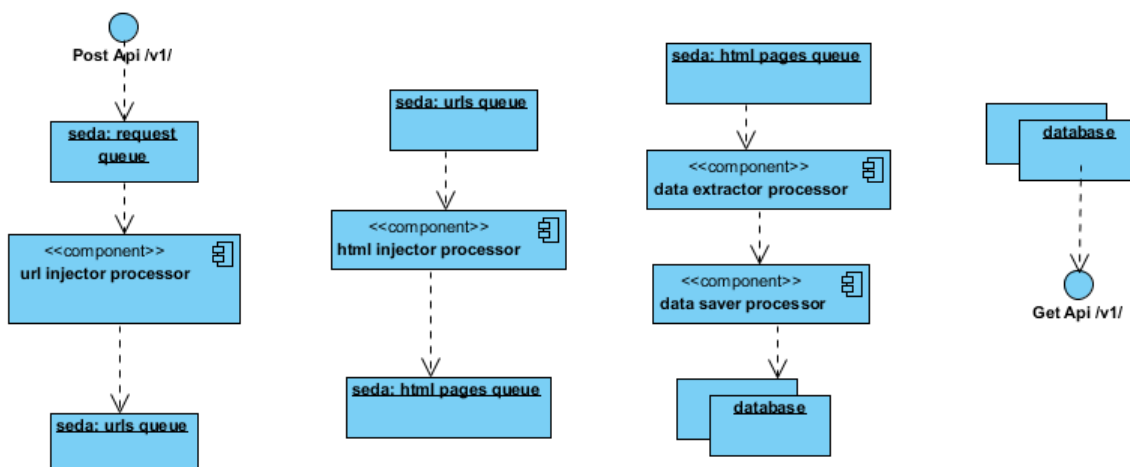


Figure 9

خزشگرها معمولاً بر پایه framework هایی مانند apache storm و nutch توسعه می‌یابند. این چارچوب‌ها امکاناتی را فراهم می‌کنند که یک کد واحد بر روی VM های متعدد اجرا شود. پیچیدگی پردازشی یک خزشگر به گونه‌ای است که یک پردازنده قوی هم نمی‌تواند نیاز ما را برطرف کند. در این پروژه از این پیچیدگی صرف نظر شده است، اما تا جای ممکن منطق کد به گونه‌ای طراحی شده تا بتواند از پس تقسیم بار پردازشی بین نخ‌ها مختلف برنامه برآید. به همین ترتیب چارچوب apache camel به کمک ما می‌آید. شمای پردازش‌های مختلفی که در شکل شماره ۹ مشاهده می‌کنید توانایی اجرا بر روی نخ‌های جداگانه را دارند. مزیت دیگر این طراحی، توانایی اجرای یک پردازش بر روی چند نخ مختلف است. برای همین عملیات دانلود صفحه‌های اینترنتی که عملیات زمانبری است می‌تواند سهم بیشتری از توان پردازشی را نسبت به بقیه عملیات‌ها داشته باشد.

## بررسی نتایج

مراحل دیپلوی این پروژه با محدودیت منابع رو به رو بود. با بالا آمدن nginx و دیپلوی قسمت frontend نیمی از رم پروژه (حدود 400MB) مصرف می‌شد. پس از بالا آمدن قسمت backend رم پروژه به مرور در اختیار پردازش‌ها قرار نمی‌گرفت و قسمت backend در محدوده زمانی مشخصی عملکرد درستی داشت.

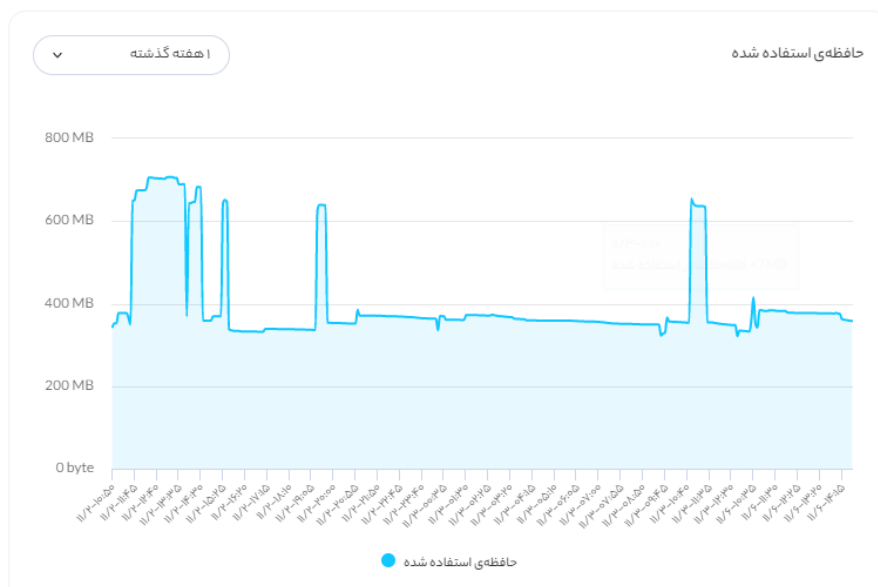
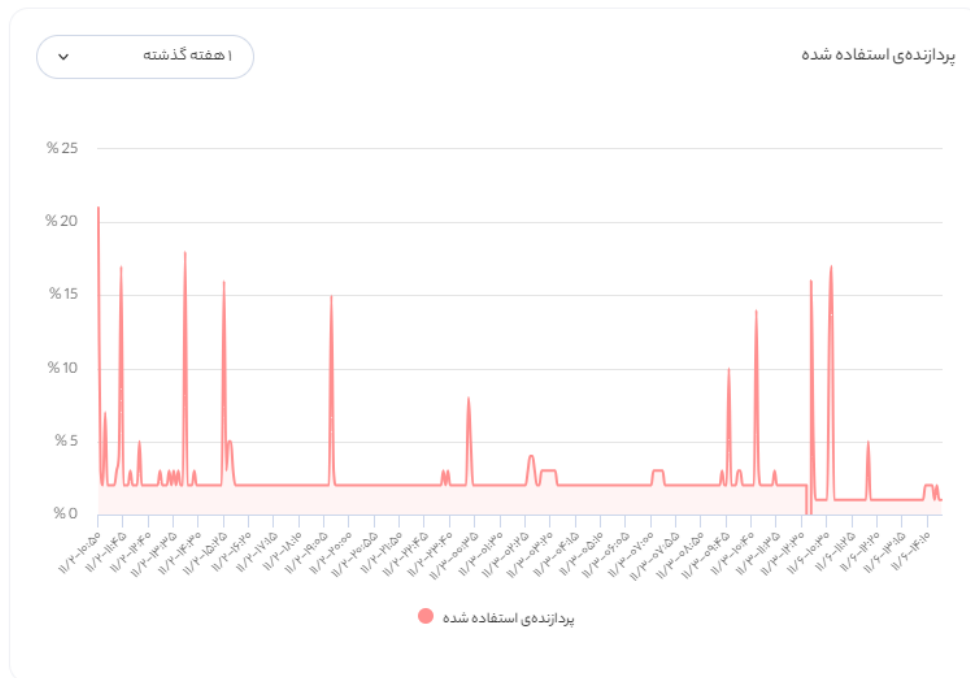
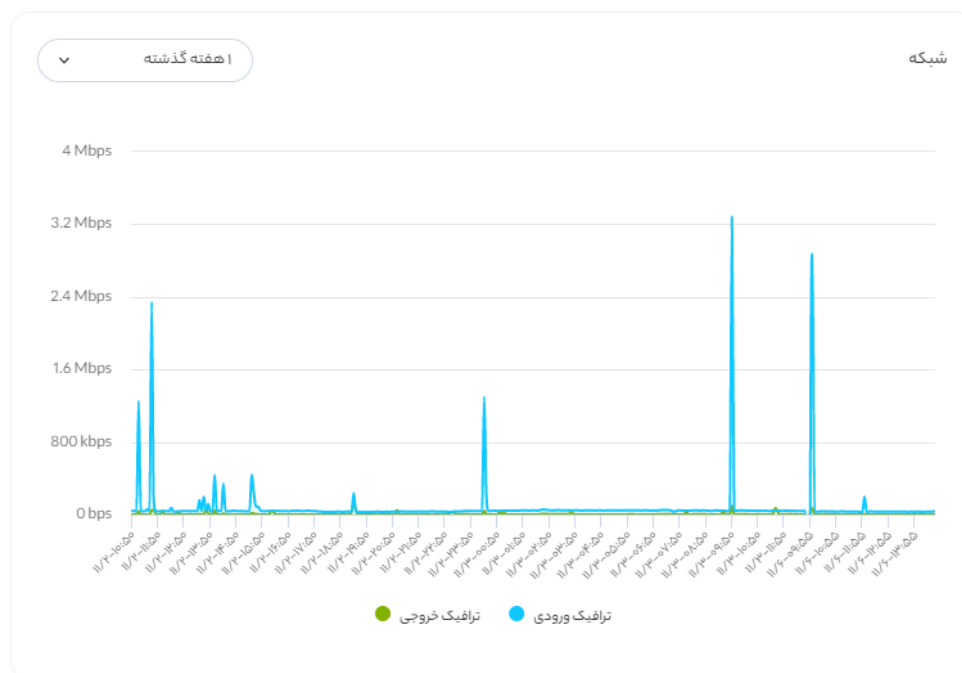


Figure 10



۱۱ Figure



۱۲ Figure

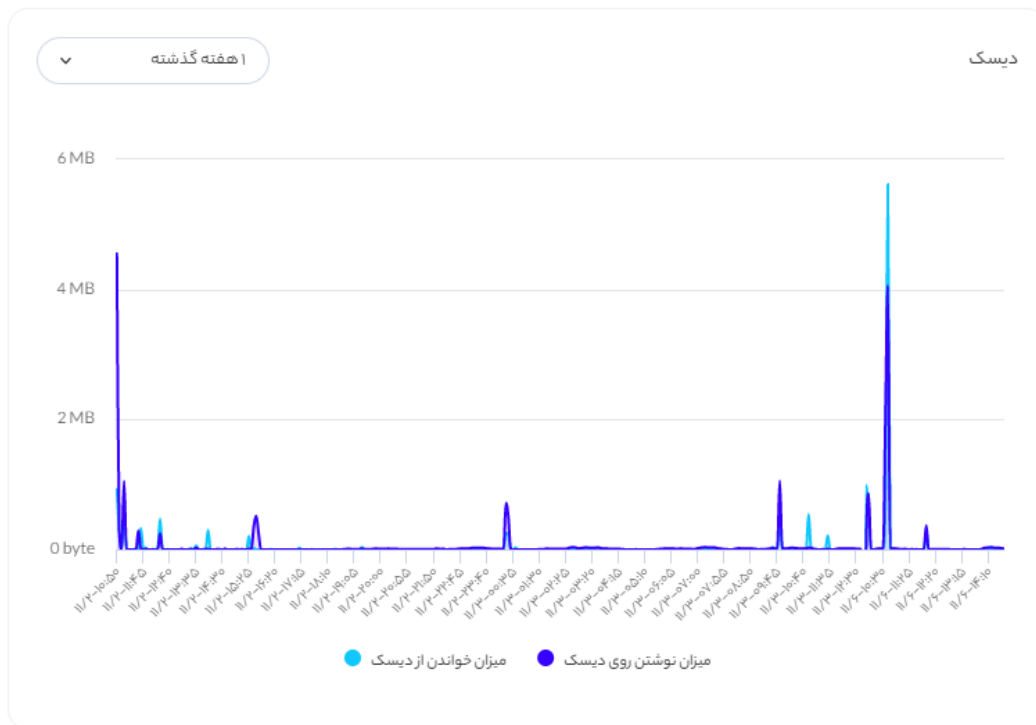


Figure ۱۳

### سناریو تست

برای قسمت تست نهایی دامنه meghdadit.ir با سایز sitemap دارای ۱۰۰۰ صفحه اینترنتی با استفاده از چهار selector خزش شد. Selectorها به شرح زیر است.

- SharedMessage\_ContentPlaceholder1\_lblMobileTitle2#
- specs > article > section:nth-child(1) > ul#
- SharedMessage\_ContentPlaceholder1\_lblItemTitle#
- specs > article > section:nth-child(3) > ul#

### پیشنهادهای

برای ادامه این پروژه اضافه شدن چهار عملکرد زیر سودمند است.

- تغییر الگوریتم url injection به صورت استخراج درختی link ها از صفحات اینترنتی
- استخراج metadata های موجود در صفحات با استفاده از کتابخانه Any23
- تغییر سیاست blockWhenFull در SedaBroker
- تهیه منابع پردازشی و ارتقای رم تا 4GB

## نتیجه گیری

جست و جو و استخراج اطلاعات همواره به عنوان یکی از نیازمندی‌های اصلی علم داده به شمار می‌روند. در وهله بعدی اگر این داده‌ها بدون ساختار و یا نیمه ساخت یافته باشند مشکلات بیشتری را به همراه دارند. این چالش‌ها عموماً دارای راه حل‌های صریح و قطعی نیستند. دانشمندان و فعالان حوزه فناوری اطلاعات با عمیق شدن و بررسی بیشتر این مشکلات می‌توانند کار مثبتی انجام دهند.

این پروژه نیز با تمام کم و کاستی‌ها در راستای همین نیاز انجام شد. امیدواریم خوانندگان این مستند با احساس همزاد پنداری با دست‌اندرکاران این پروژه در راستای حل مشکلات جامعه و درک صحیح آنها گام بردارند.

## منابع

۱. <https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle>
۲. <https://camel.apache.org/docs>
۳. <https://reactjs.org/docs/getting-started.html>
۴. <https://camel.apache.org/components/3.20.x/seda-component.html>
۵. <http://stormcrawler.net>