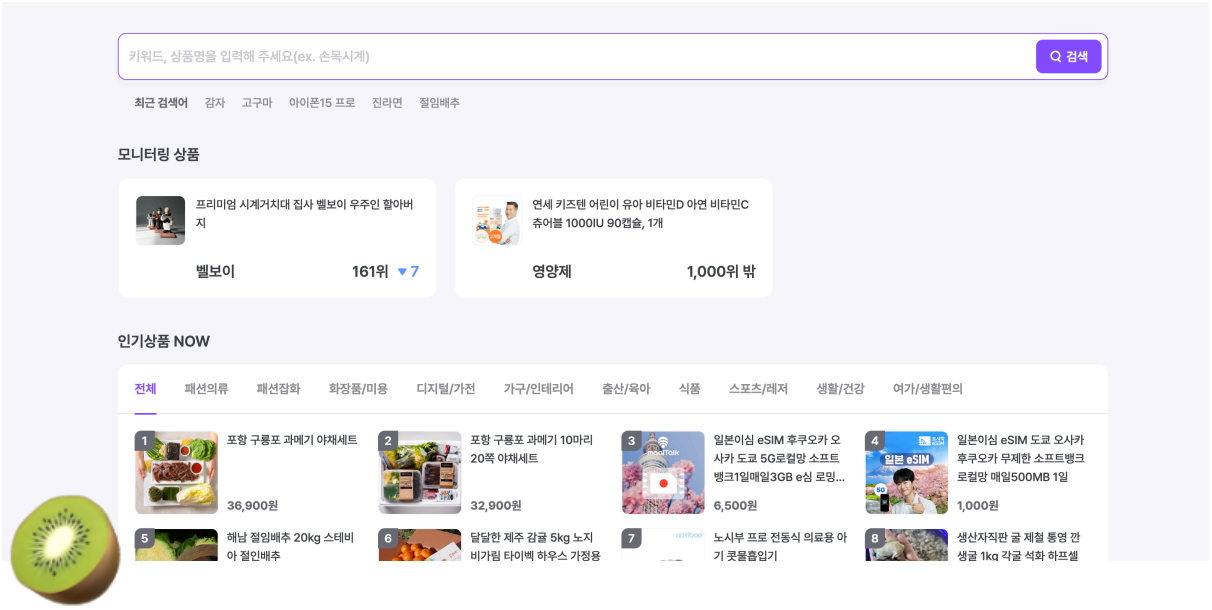


셀러키워드(셀키) 1.7 고도화

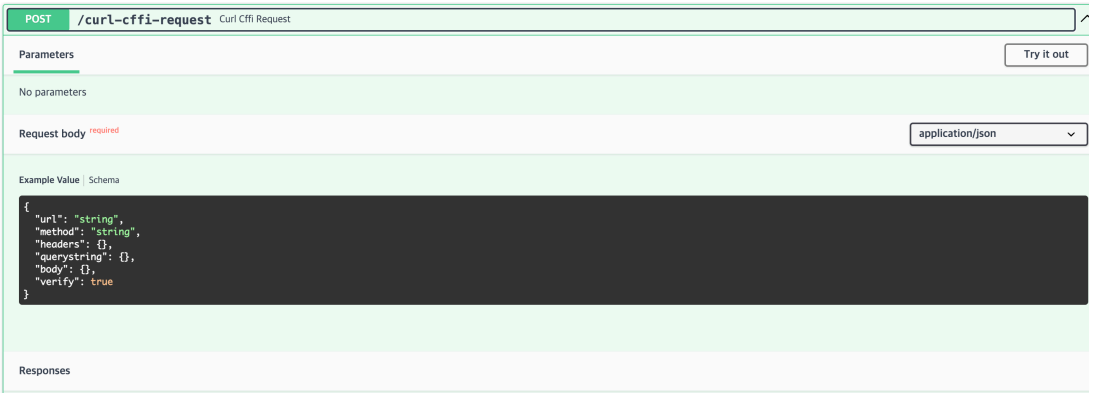
태그	AWS Crawling 데이터엔지니어 서클플랫폼 자체개발
날짜	@2023년 12월 1일 → 2024년 3월 1일

서비스

- 🔗서비스 링크
- 제공 서비스
    - 여러 오픈마켓의 데이터를 한군데 모아 소비자가 아닌 판매자를 위한 형태로 가공하여 판매전략에 도움을 주는 서비스 입니다.

서비스에서 맡은 업무

1. 네이버 쇼핑 데이터 수집 및 스크래핑 방지 회피 로직 개발
    - Request 요청을 중개 할 수 있는 Serverless Proxy 서버를 구성하였습니다.



    - 또한 봇탐지를 방지하기 위해 여러가지 조치를 취했습니다.
        - **curl-cffi 라이브러리** 사용 - 링크: https://github.com/lexiforest/curl_cffi

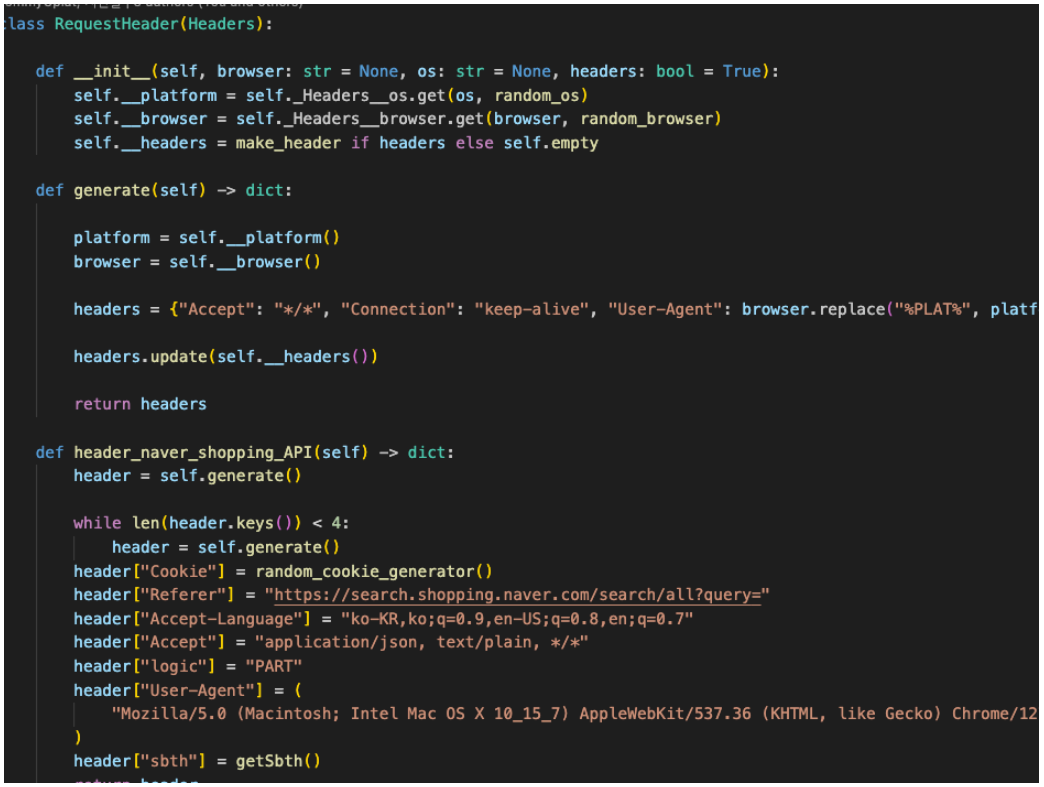
> 👉 curl-cffi 는 스크래핑 요청이 마치 chrome, safari 등의 브라우저 처럼 보이게 만들어서 서버 차단을 우회할 수 있도록 합니다.

        - **자체 개발 CookieGenerator 운영** - Playwrights 를 사용하여 동적크롤링을 통해 실제 Cookie 발급

```python
def random_cookie_generator():
    cookie_list = [
        [redacted]
    ]
    NNB_list = [
        cookie[cookie.find("NNB=") : cookie.find(";", cookie.find("NNB="))]
        for cookie in json.loads(os.environ["DATALAB_KEYWORD_COOKIE_LIST"])
    ]
    shuffle(cookie_list)
    choice_cnt = randint(3, 6)
    cookie = cookie_list[:choice_cnt]
    cookie.append(f"{NNB_list[randint(0, len(NNB_list)-1)]};")
    shuffle(cookie)
    return " ".join(cookie)
```

> 과정은 다음과 같습니다.
>
> 👉
>
> 1. 필요한 Cookie 리스트 분석
> 2. 매일 배치를 통해 동적크롤링으로 Cookie 리스트를 저장
> 3. 실제 라이브에서는 새로운 Cookie 정보로 안정적인 스크래핑

- Fakeheaders 라이브러리를 현재 운영 방식에 맞게 수정하여 **헤더 정보를 랜덤하게 생성**

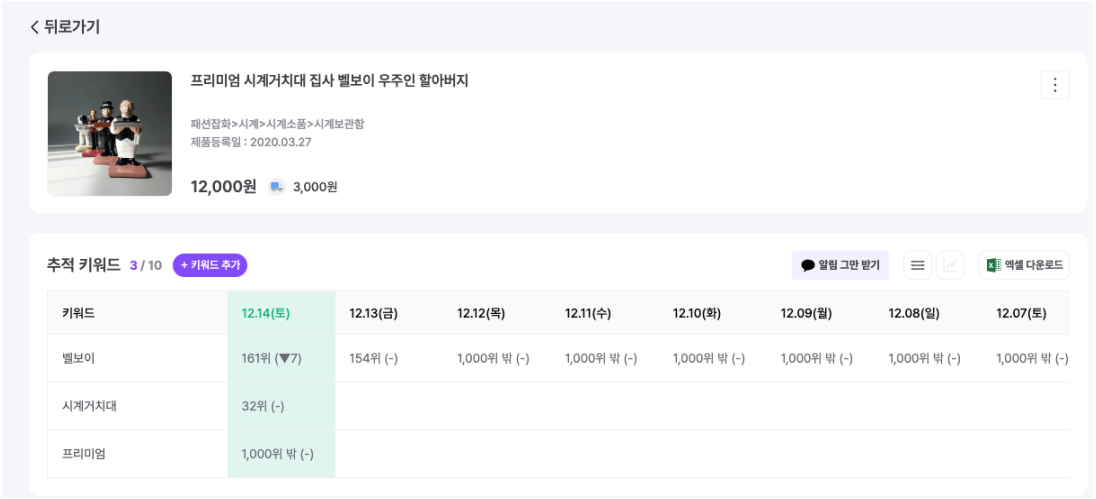


```python
class RequestHeader(Headers):

    def __init__(self, browser: str = None, os: str = None, headers: bool = True):
        self.__platform = self._Headers__os.get(os, random_os)
        self.__browser = self._Headers__browser.get(browser, random_browser)
        self.__headers = make_header if headers else self.empty

    def generate(self) -> dict:

        platform = self.__platform()
        browser = self.__browser()

        headers = {"Accept": "*/*", "Connection": "keep-alive", "User-Agent": browser.replace("%PLAT%", platf

        headers.update(self.__headers())

        return headers

    def header_naver_shopping_API(self) -> dict:
        header = self.generate()

        while len(header.keys()) < 4:
            header = self.generate()
        header["Cookie"] = random_cookie_generator()
        header["Referer"] = "https://search.shopping.naver.com/search/all?query="
        header["Accept-Language"] = "ko-KR,ko;q=0.9,en-US;q=0.8,en;q=0.7"
        header["Accept"] = "application/json, text/plain, */*"
        header["logic"] = "PART"
        header["User-Agent"] = (
            "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/12
        )
        header["sbth"] = getSbth()
```

> 👉 특정 URL 은 Cookie 뿐만 아니라 Referer 또는 **User-Agent 의 Chrome 버전에 민감한 경우**도 있어 전용 Headers 를 `random_cookie_generator` 와 같이 사용할 때도 있습니다.

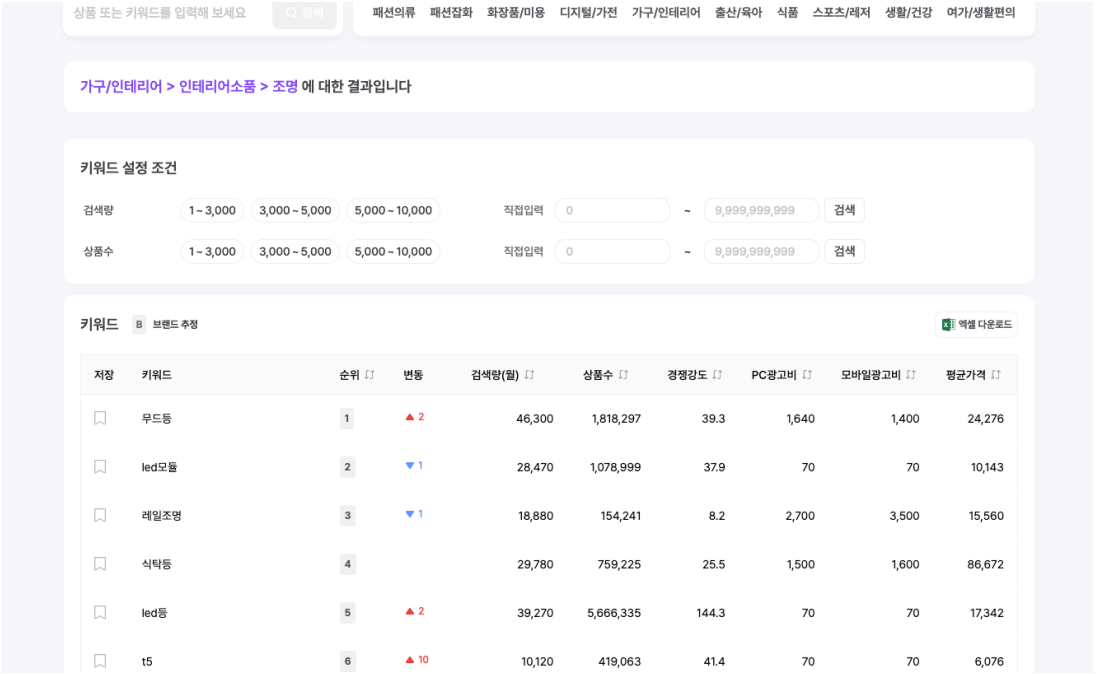
2. FastAPI 도입 및 로직 비동기화
    - 셀키 1.7 에서는 FastAPI 로 변경 및 수집 로직을 비동기화 하여 속도를 대폭 향상 시켰습니다.
3. Airflow 를 통한 키워드 모니터링, 키워드 탐색 - Daily, Monthly 스케쥴링

- 키워드 모니터링



👉 네이버쇼핑에서 **특정 상품이 특정 키워드 검색기준으로 몇페이지에 몇 순위로 노출**되고 있는지**매일 스크래핑하여 순위를 계산하는 서비스** 입니다.

- 키워드 탐색



👉 월별로 인기 키워드에 대한 검색량, 상품수, 네이버쇼핑 평균 가격등을 스크래핑합니다.
→ 카테고리별로 인기키워드에 대해 분석서비스를 제공합니다.

개발후기

> 기존에 토이프로젝트나 간단한 프로젝트에 데이터가 필요하여 크롤링할때와 유료 서비스로서 여러 크롤링을 `실패없이`, `빠르게`, `대량으로` 이 3가지를 서비스 한다는것은 차원이 다르다는 것을 깨달았습니다.
> **하나의 화면에도 많게는 7번의 스크래핑을 해야하는데 7개를 지속적으로 빠르게 수집해야 하는 비동기 로직**과 **자체구성한 Proxy 서버**, **여러 크롤링 소스 분석**등의 일을 하면서 정말 새벽까지도 일하고는 했었는데요. ***서비스 차원에서의 크롤링은 차원이 다르다는것을 배우는 프로젝트***가 되었습니다.

☐ 트래픽

cnt	도메인	method	type	a/sync	base_url	비고
2	네이버 광고 API	POST	json	async	https://api.searchad.naver.com	발급한 API key 사용
1	데이터랩	POST	json	async	https://datalab.naver.com/qcHash.naver	trenResult.naver 용 Hash key 발급용
1	데이터랩	GET	json	async	https://datalab.naver.com/keyword/trendResult.naver	
1	네이버 API	GET	json	async	https://api.naver.com/keywordstool	
1	네이버 검색	GET	text	async	https://search.naver.com/search.naver	html 파싱
2	네이버쇼핑	GET	json	async	https://search.shopping.naver.com/api	네이버전체, 네이버페이 한 번씩
1	네이버쇼핑	GET	text	async	https://search.shopping.naver.com/search/all	네이버 연관 키워드

키워드 분석화면을 위해 필요한 스크래핑 수

각 화면구성에 필요한 API 리스트 문서

👉 각 화면을 구성하며 해당 페이지의 요구사항을 채우기 위해 여러 사이트를 탐색하며 크롤링 가능성 분석등을 진행하는데 시간가는지 모르고 재밌게 했던 것 같습니다. 😂