웹 크롤링을 이용한 미세먼지 문제 대응 어플리케이션

A fine dust response application using web crawling

김호균 (Ho-gyun Kim) [1]

[1] 한국외국어대학교, 컴퓨터공학부; twink7753@naver.com

한글 요약:

본 연구에서 구현한 실시간으로 미세먼지(PM10)와 초미세먼지(PM2.5)의 농도 데이터를 수집하고 지역별, 연령별 건강 수칙을 확인할 수 있는 모바일 어플리케이션은 한국환경공단(에어코리아)에서 실시간으로 업데이트 및 제공하는 국내 대기오염 측정망에서 측정된 데이터들을 웹 크롤링 시스템을 통해 수집 및 정제하여 사용자가 원하는 정보를 제공해주고 사용자에게 알맞은 건강수칙 등의 솔루션을 제공한다. 한국환경공단(에어코리아) 에어코리아 사이트(https://www.airkorea.or.kr)에서 제공하는 시도별 대기정보 웹 페이지의 url 패턴을 분석하고 크롤링한 후에 클라이언트와 구축된 restful api 서버를 통해 통신을 하는 방식을 사용하였다. 이 과정에서 매시간마다 모든 웹페이지를 크롤링하는 방식과 클라이언트의 요청이 올 때마다 필요한 웹페이지만 크롤링하는 방식, 이 두가지 방식으로 크롤링 시스템을 구축하고 비교하였다. 본 연구자는 개발 과정 중 웹 크롤링 기법과 서버구축에 초점을 맞춰 연구를 진행하였다.

핵심어: 웹 크롤링, RESTful api, 안드로이드 어플리케이션

영문 요약: In this study, Mobile application that can collect concentration data of fine dust(PM10) and ultrafine dust(PM2.5) in real time and check health rules by region and age collects and purifies data measured by the domestic air pollution measurement network updated and provided by the Korea Environment Coporation(Airkorea) in real time through a web crawling system. In this application, communication with the client through restful api server was used after analyzing and crawling the url pattern of the standby information web page for each city provided by Air Korea web page (https://www.airkorea.or.kr). In this process, the crawling system was built and compared in two ways: crawling all web pages every hour and crawling only the necessary web pages every time a request comes from clients. During the development process, this researcher focused on web crawling techniques and server construction.

Keywords: web crawling , RESTful api, android application

본 논문은 2021 학년 2 월에 제출된 한국외국어대학교 컴퓨터공학부 졸업논문이다. 2021.12.05

지도교수: 신현수

서명:

참여한 캡스톤 설계 (해당자만)

설계명:

1. 서론 – Introduction

1.1 연구 동기

갈수록 심해지는 대한민국의 미세먼지 문제에 비해 대중에게 제공되는 정보가 너무 적거나 접근성이 좋지 않는 경우가 많기 때문에 실질적으로 미세먼지 문제에 있어서 건강의 위협에 직접적으로 노출되어 있다. 선행된 연구 결과에 따르면 초미세먼지(PM2.5) 농도가 10g/m^3 씩 증가할 때마다 총 사망률이 14% 증가하며, 심혈관 호흡기계 사망률은 19% 증가한다고 보고되었다. 또한, 미세먼지에 따른 한국인의 질병 부담에 대해 조사한 결과, 폐암 등 폐질환 뿐만 아니라, 아토피, 모발 및 두피 손상, 결막염 등 피부질환과 안질환에도 영향을 미치는 것으로 보고되었다[1]. 이와 같이 미세먼지는 인간의 삶의 질에 여러 방면에서 영향을 끼치는 요인이나, 미세먼지의 발생 자체의 억제는 현재 사실상 통제 불가능한 요인이기 때문에 미세먼지에 대한 개인의 대처가 가장 중요한 부분이라고 할 수 있다.

공공의 건강 증진을 위해 매시간마다 업데이트 되는 미세먼지(PM10), 초미세먼지(PM2.5)의 농도를 쉽게 확인을 할 수 있도록 하고 실질적으로 일반인의 입장에서 건강을 보호할 수 있도록 지역별, 연령별의 기준으로 세분화하여 사용자에게 알맞은 건강 수칙 등의 솔루션을 제공하는 모바일 어플리케이션을 구현하는 것이 목표이다.

1.2 개발 구상

1.2.1 어플리케이션의 전체 구성도

본 연구에서 구현할 모바일 어플리케이션은 현재 지역의 미세먼지(PM10)와 초미세먼지(PM2.5)의 농도를 볼 수 있는 메인 페이지와 평균, 최대, 최소, 시간별 농도 변화 등 상세 정보를 확인할 수 있는 페이지, 알맞은 미세먼지 대응법을 제공해주는 페이지로 구성된다(그림 1).

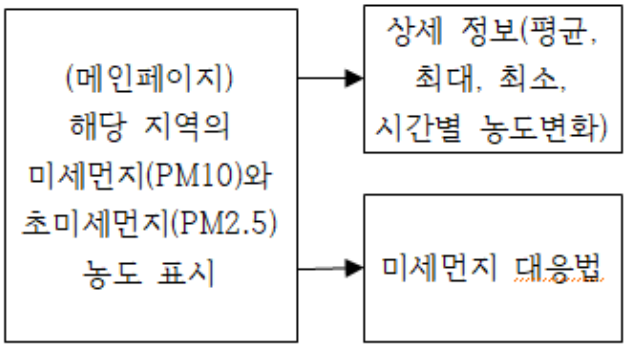


그림 1. 모바일 앱 구성도

1.2.2 웹 크롤링 방식

어플리케이션에서는 지역별, 미세먼지 종류별 농도 값을 필요로 한다. 본 연구에선 한국환경공단(에어코리아, https://www.airkorea.or.kr)에서 제공되는 농도 측정값을 크롤링하여 데이터화 시킨다. 에어코리아에서는 전국 측정소별로 매시간마다 측정 데이터들을 테이블형태의 HTML 웹페이지로 업데이트 하고 있는데 한 개의 지역에서 측정되는 한가지 미세먼지(미세먼지 또는 초미세먼지)의 농도 값을 한 웹 페이지에 제공해준다. 이에 측정 값들을 파이썬(python)의 bs4(beautifulsoup4)모듈을 사용하여 HTML 소스분석(HTML parsing) 기법으로 추출 및 정제한다.
본 연구에서는 두가지의 크롤링 실행방식을 제시하며 방식은 다음과 같다.

1. 클라이언트가 서버에 요청을 할 때 마다 해당 사이트를 크롤링
2. 매시간마다 모든 웹 페이지 크롤링

두가지 방식 중 크롤링 대상 웹 사이트 연결 횟수와 클라이언트의 요청에서부터 서버의 응답까지 걸리는 시간을 고려하여 더 효율이 뛰어난 방법을 채택한다.

1.2.3. 모바일 어플리케이션 개발

Android OS 를 타겟으로 하여 Java 기반의 Android Studio 를 이용하여 어플리케이션을 개발한다. 어플리케이션에서는 각 수치 및 수치 별 위험도를 알리며, 해당 위험도에서 일반인의 입장에서 어떤 방법으로 미세먼지로 인한 피해를 최소화할 수 있을지에 대한 대처 방법을 표시하도록 한다. 어플리케이션 사용자의 지역별, 연령별, 직업별 등의 정보를 받아 각 경우의 수에 대한 솔루션을 세분화하여 제공하도록 한다. 이와 같은 개인정보들의 입력은 선택사항으로 한다.

1.2.4. 앱과 서버간 데이터 전송 방식

본 연구에서 구현한 앱에서는 농도의 정보 데이터만을 필요로 하고 다른 데이터들을 서버에 올리거나 받는 과정이 필요가 없다. 간단한 GET 요청만 수행할 수 있는 restful api 를 구축하여 클라이언트가 요청에 맞게 데이터를 json 형식으로 반환한다.

2. 연구 방법 및 결과

2.1. 웹 크롤링(web crawling)시스템

2.1.1. 웹 크롤링의 정의

웹 클롤링이란 웹 사이트에서 정보 자원을 자동화된 방법으로 수집, 분류, 저장하는 것이다. 크롤링 대상 페이지를 웹 클라이언트를 통해 접속하고 접속된 웹 클라이언트를 통해 HTML 파일을 파싱(Parsing)하여 필요한 데이터를 원하는 만큼 가져오는 과정이다. 웹 크롤러는 웹 크롤링을 하는 프로그램으로 대체로 방문한 사이트의 모든 페이지의 복사본을 생성하는데 사용되며, 검색 엔진은 이렇게 생성된 페이지를 보다 빠른 검색을 위해 인덱싱한다. 또한 크롤러는 링크 체크나 HTML 코드 검증과 같은 웹 사이트의 자동 유지 관리 작업을 위해 사용되기도 하며, 자동 이메일 수집과 같은 웹 페이지의 특정 형태의 정보를 수집하는데도 사용된다. 일반적인 크롤러는 대개 시드(seeds)라고 불리는 URL 리스트에서부터 시작되는데 페이지의 모든 하이퍼링크를 인식하여 URL 리스트를 갱신한다[2]. 갱신된 URL 리스트는 재귀적으로 다시 방문한다. 크롤러의 일반적인 형태와 구조는 아래[2]와 같다. (그림 2)

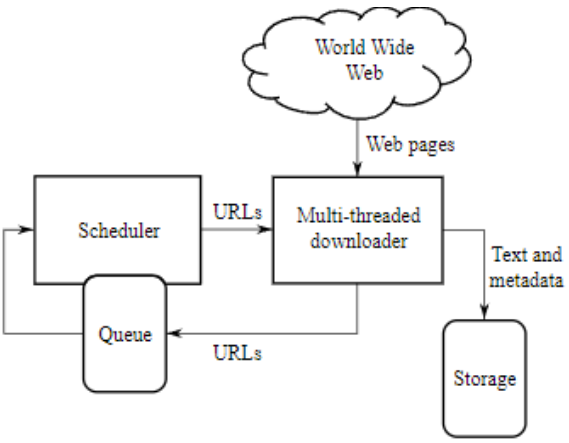


그림 2. 크롤러의 형태 [출처] 위키백과

2.1.2 데이터 추출

어플리케이션 구성을 위해 실시간으로 업데이트 되는 미세먼지/초미세먼지 농도 정보가 필요하고 이를 스크래핑할 필요가 있다. 에어코리아(www.airkorea.or.kr)에서는 전국 측정소별로 매시간마다 자료를 테이블형태로 업데이트 하고 있다. 대기오염정보의 open-api 또한 제공하고 있는데 본 연구에서는 HTML 의 태그들을 파싱해서 데이터를 추출할 것이기 때문에 open api 는 이용하지 않는다. 다른 많은 정보를 필요로 하지 않고 테이블에서의 값들만 필요하다. 따라서 Python 을 기반으로 하는 Beautifulsoup4 라이브러리를 이용하여 해당 사이트를 HTML 소스 분석(HTML parsing)하여 필요한 데이터를 추출하였다.

에어코리아는 에어코리아(www.airkorea.or.kr) > 실시간 자료조회 > 시도별 대기정보 > (시도별)상세자료 에서 매 시간마다 측정 농도를 업데이트 해준다. 정보를 추출하는데 있어서 시도별,

미세먼지 종류별 제공되는 URL 이 다르고, 매시간(업데이트 주기)마다 테이블에 새로 업데이트 되는 농도를 확인할 수 있는 URL 이 다르므로 URL 을 분석하여 패턴화 시킬 필요가 있다. 크롤링 할 사이트의 url 을 완성시키기 위해서 필요한 parameter 로는 대기오염물질종류코드, 현재날짜, 시간코드, 지역코드, 월 초의 날짜, 해당 월의 일수 가 필요하다. 대기오염물질종류코드와 지역코드는 클라이언트로부터 받을 지역 이름과 대기오염물질 이름을 미리 준비해 놓은 dictionary 를 통해 얻을 수 있고 나머지 매개변수들은 datetime 모듈과 calendar 라이브러리의 monthrange 모듈을 사용하여 얻을 수 있다. 패턴화된 URL(그림 3)과 url 완성을 위해서 필요한 코드의 내용(그림 4)은 아래와 같으며 완성된 url 의 예시로는 2021 년 11 월 5 일 서울지역 PM2.5 농도의 20 시에 업데이트된 현황을 확인하기 위한 url 주소는 다음과 같다.

- https://www.airkorea.or.kr/web/sidoAirInfo/sidoAirInfoDay01?itemCode=10008&ymd=2021-11-05%2019&areaCode=02&tDate=2021-11-01&monthDay=30

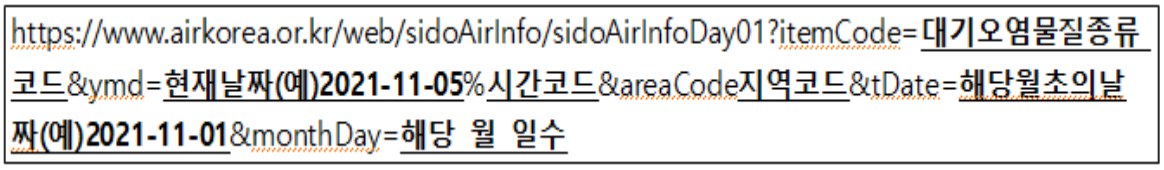


그림 3. 크롤링 대상 url pattern.

```
area_code = {'서울': '02', '부산': '051', '대구': '053', '인천': '032',
             '광주': '062', '대전': '042', '울산': '052', '경기': '031', '강원': '033',
             '충북': '043', '충남': '041', '전북': '063', '전남': '061', '세종': '044',
             '경북': '054', '경남': '055', '제주': '064'}
item_code = {'PM2.5': '10008', 'PM10': '10007'}
```

```
hour_code = hour + 1999
```

그림 4. url 완성을 위해 필요한 코드들

완성된 url 로 GET 으로 request 하여 해당 사이트의 HTML 정보를 얻고 이 HTML 을 bs4(beautifulsoup4)모듈로 HTML 소스 분석(HTML parsing)하여 필요한 정보들을 추출한다. 해당 사이트의 HTML 소스를 분석해보면 크롤링할 데이터 값들은 <table> -> <tbody> -> <tr> -> <th>태그 안에 있으며(그림 5-a) ' ', '₩r;', '₩n' 등의 제어문자 같은 필요 없는 값들을 제거 하고 정보만을 추출하여 이중리스트 형태(그림 5-b)로 반환할 수 있게 데이터를 수집하였다.

그림 5. 데이터의 크롤링하기 전과 후 (a) table태그의 구조 (b) 결과 자료형

2.1.3 크롤링 시스템 구축

본 연구에서는 크롤링 시스템을 다음의 두가지 방식으로 구현 및 비교를 하였다.

방법 1. 클라이언트로부터 요청이 올 때마다 필요한 웹페이지만 크롤링하는 방법.

방법 2. 클라이언트로의 요청과 관계없이 매시간마다 모든 웹페이지를 크롤링하는 방법.

1 번 방법으로 구현한 크롤링 시스템의 전체적인 동작 방식을 정리하면 아래의 그림과 같다(그림 6). 클라이언트에서 필요한 정보(지역이름, 대기오염물질이름)를 요청하면 준비된 코드화 프로그램을 통해 정보의 코드화(지역명 코드, 대기오염물질 코드)가 이루어지며 요청하는 현재 시간 정보 역시 코드화가 이루어진다. 만들어진 코드들의 조합으로 크롤링할 사이트의 url 패턴이 완성된다. bs4(beautifulsoup4)모듈을 통해 해당 사이트의 HTML 소스 분석을 진행하며 필요한 데이터를 추출하고 알맞은 형태로 정제 후 반환한다.

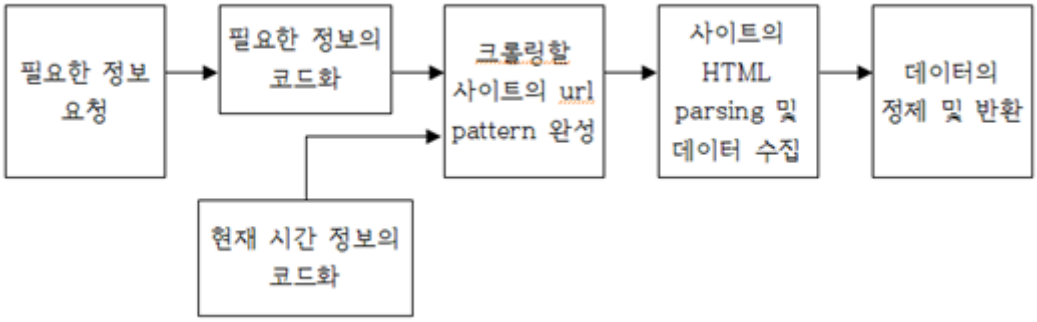


그림 6. 필요한 웹페이지만 크롤링하는 크롤링시스템 프레임워크

2 번 방법으로 구현한 크롤링 시스템의 프레임워크는 아래의 그림과 같다(그림 7). 앞에서 다룬 1 번 방법처럼 클라이언트의 요청을 기다리는 것이 아니고 정해진 시간에 맞춰 매시간마다 전체 사이트를 대상으로 크롤링을 한다. 이 과정에서 크롤링을 반복하기 위해 apscheduler 모듈을 사용한다. 에어코리아에서 측정값들을 매시 정각에 업데이트가 이루어지는 것이 아니기 때문에 누락되는 데이터가 없게 하기 위해서 매시 20 분에 반복실행 할 수 있도록 한다. 데이터 추출 방법은 동일하며 restful api 서버의 자원에 맞게 데이터들을 정제하고 서버에 올려 둔다. 이후 클라이언트로부터 요청이 들어오면 서버에 미리 올려 놓았던 데이터를 반환한다.

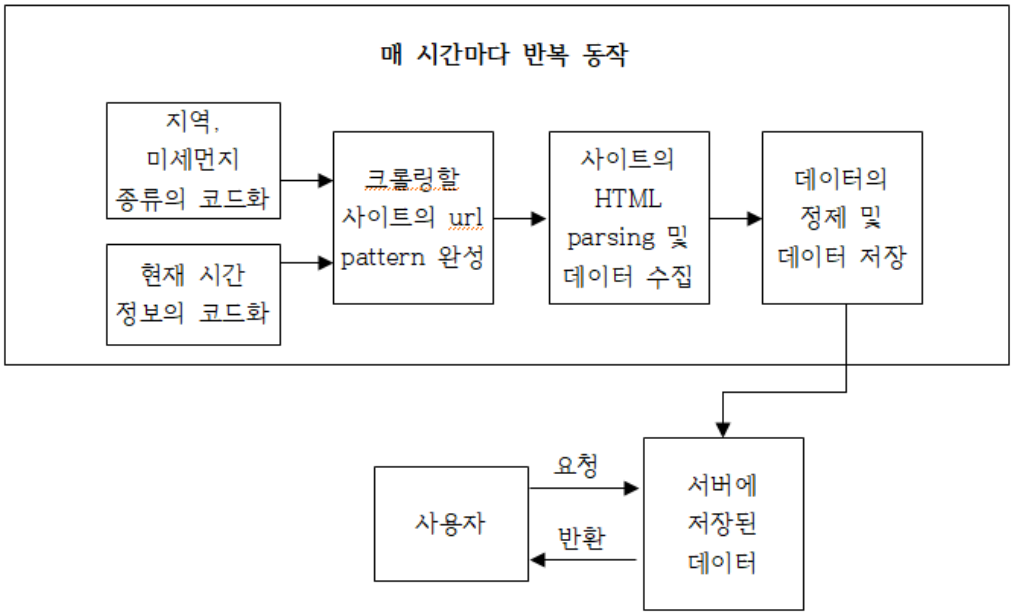


그림 7. 자동화된 크롤링시스템 프레임워크

2.2. RESTful api 서버 구축

2.2.1. RESTful api 의 정의

RESTful api 란 REST 아키텍처의 제약조건을 준수하는 어플리케이션 프로그래밍 인터페이스를 뜻한다. 여기서 REST 는 Representational State Transfer 의 줄임말이다[3].

api 는 애플리케이션 소프트웨어를 구축하고 통합하는 정의 및 프로토콜 세트이다. 때때로 api 는 정보 제공자와 정보 사용자 간의 계약으로 지칭되며 소비자에게 필요한 호출(request)과 생산자에게 필요한 응답(response)을 구성한다. 즉, 컴퓨터나 시스템과 상호작용하여 정보를 검색하거나 기능을 수행하고자 할 때 api 는 사용자가 원하는 것을 시스템에 전달할 수 있게 지원하여 시스템이 이 요청을 이해하고 이행하도록 할 수 있다. api 는 클라이언트, 클라이언트가 얻으려 하는 리소스 사이의 조정자이다. Restful api 를 통해 요청이 수행될 때 restful api 는 리소스 상태에 대한 표현을 요청자에게 전송한다. 이 정보 또는 표현은 HTTP: JSON, HTML, XLT 또는 일반 텍스트를 통해 몇 가지 형식으로 전송된다. JSON 은 그

이름에도 불구하고 사용 언어와 상관이 없을 뿐 아니라 인간과 머신이 모두 읽을 수 있기 때문에 가장 널리 사용된다[4,5,6].

쉽게 말해서 REST api 는 웹에서 사용되는 데이터나 자원(Resource)을 HTTP URI 로 표현하고, HTTP 프로토콜을 통해 요청과 응답을 정의하는 방식을 말한다.

2.2.2. RESTful api 서버의 구현

먼저 서버에서 동작할 크롤링 시스템을 python 으로 구현하였기 때문에 서버 또한 python 으로 구현하였다. python 으로 서버를 구현할 수 있는 웹 프레임워크(Web framework)는 크게 Django 와 Flask 가 있다. 그 중 Flask 는 마이크로 웹 프레임워크(Micro Web Framework)이며 현실적으로 서버의 어플리케이션단을 구현하기보단 api server 의 역할을 더 많이 한다[7]. 모바일 어플리케이션에서 필요한 정보를 조회하는 용도로서만 서버를 구현하기 때문에 GET 요청만 수행할 수 있으면 된다. 따라서 다른 수행능력(POST, PUT, DELETE 등)의 구현이 필요하지 않기 때문에 데이터베이스를 연동하지 않았고, 무거운 Django 라이브러리 보다는 가볍고 GET 요청만 구현할 수 있는 Flask 라이브러리를 이용하고 flask-restx 모듈을 사용하여 서버를 구현하였다. 클라이언트로부터 GET 요청을 받게 되면 서버에서 크롤링하여 수집 및 정제한 데이터를 json 형태로 반환한다(그림 8.).

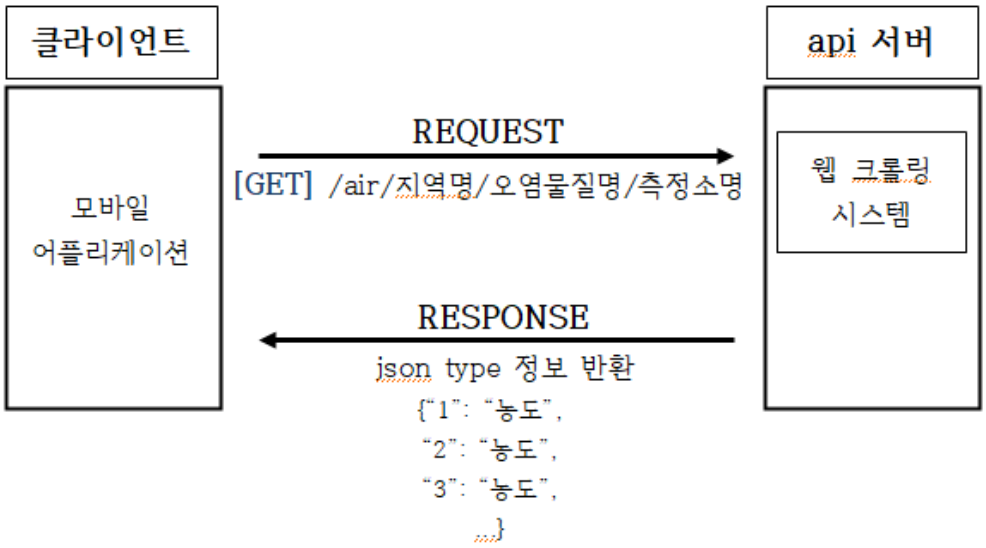


그림 8. 클라이언트와 api 서버간의 통신형태

2.1.3 에서 구현한 크롤링 프레임워크의 형태에 따라 구조가 달라지는데 서버를 구성하는 요소들은 다음과 같다.

① 2.1.3 -방법 1 의 경우

2.1.3 – 방법 1 을 사용할 경우 요청이 들어올 때마다 그에 맞는 사이트를 크롤링하여야 하기 때문에

해당 사이트의 url 을 찾을 수 있어야한다. 따라서 크롤링 대상 URL 에서의 지역 코드와 오염물질의 코드를 RESTful api 서버에서의 자원이름과 일치시키는 방법을 사용하여 클라이언트에서 요청된 지역과 오염물질의 범위 내 사이트만 크롤링할 수 있게 하였다. (그림 9)

그림 9 . RESTful api 서버의 자원과 크롤링할 사이트의 URL 매개변수의 일치화

② 2.1.3 -방법 2 의 경우

2.1.3- 방법 2 에서는 클라이언트의 요청과는 관계없이 매시간마다 모든 사이트에 대해 크롤링을 하고 정제된 데이터들을 서버에 올려놓는다. 서버에 올려져 있는 데이터들을 세분화하여 필요한 데이터들을 반환할 수 있도록 한다. 크롤링 시스템에서의 반환 데이터 형식은 이중리스트 형태이다. 따라서 클라이언트의 요청을 받고 데이터를 반환하기 전에 json 형태로 바꾸어 반환할 수 있도록 설계하였다

클라이언트가 필요한 자원을 다음의 네 가지 방법으로 요청할 수 있다.

1. GET | air/<string:sido>/<string:item>
2. GET | air/<string:sido>/<string:item>/<string:location>
3. GET | air/<string:sido>/<string:item>/<string:location>/<int:hour>
4. GET | location

1 번에서는 지역 명(sido)과 대기오염물질의 이름(item)의 자원을 요청하면 당일 전체 시간에 대한 농도를 반환하고 2 번은 측정소 이름(location)을 추가로 자원으로 요청하면 해당 측정소의 하루(1 시~24 시)의 모든 농도 값을, 3 번에선 시간(hour)을 추가로 요청해서 해당 측정소의 해당 시간 농도 값을 반환하도록 한다. 반환하는 자료형은 다음과 같다 (그림 10-1).

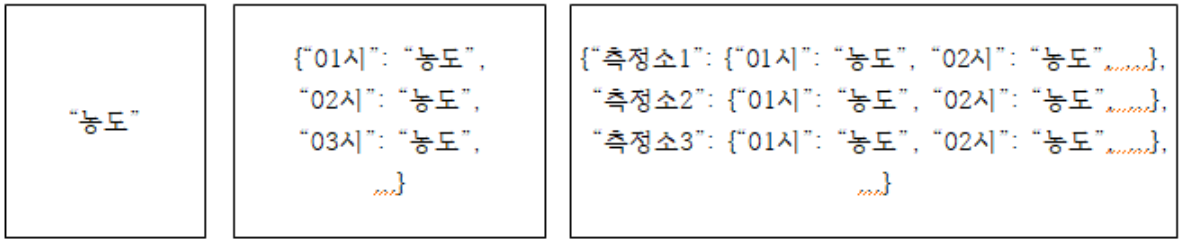


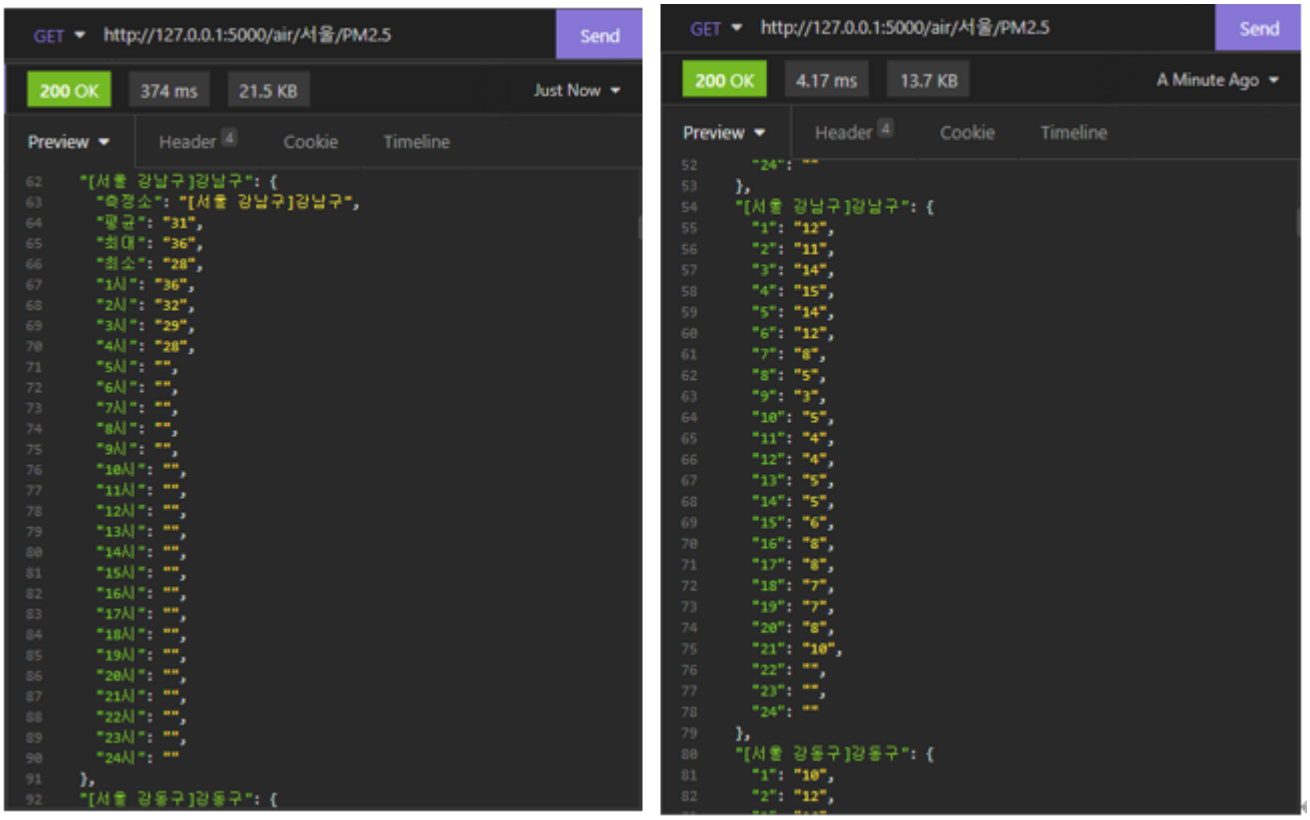
그림 10-1. 서버에서 반환되는 json 자료형(1,2,3 번 방식)

모바일 앱에서 측정소 이름들을 스피너(spinner)로 보여주고 사용자가 원하는 지역을 선택할 수 있게 하기 위해서 4 번의 형식으로 요청하면 전국 모든 측정소의 이름들의 배열을 반환하고 자료형은 다음과 같다(그림 10-2).

{"location": ["측정소1", "측정소2, "측정소3",.....]
}

그림 10-2. 서버에서 반환되는 json 자료형(4 번 방식)

서버가 잘 동작하는지 확인하기 위해 클라이언트를 대신해 줄 수 있는 "Insomnia" 프로그램을 사용하여 서버로 GET 요청을 보내고 응답까지의 걸리는 시간과 데이터의 반환이 잘되는 것을 확인하였다(그림 11).



(a) (b)

그림 11. Request 와 response 의 결과
(a) 2.1.3-1 번 방식의 request, (b) 2.1.3-2 번 방식의 request

3. 결과 분석

본 연구에서는 매시간마다 정보를 수집할 수 있도록 자동화된 크롤링 시스템을 설계하는 방법과 클라이언트의 요청이 올때마다 요청에 맞는 지역, 미세먼지 종류 등에 맞는 웹 페이지만 크롤링 하는 방법의 두가지 크롤링 시스템 프레임워크를 제시하였다. 두가지의 시스템을 크롤링 할 웹 사이트 연결 횟수와 클라이언트의 요청에서부터 서버의 응답까지 걸리는 시간을 고려하여 비교해보았다.

① 크롤링 대상 사이트 연결 횟수 비교

자동화된 크롤링 시스템에서는 1 시간에 1 번 전체 웹 사이트를 크롤링한다. 이 때 연결(GET 요청)횟수는 34 번 (17(시도별 분류한 개수)x2(미세먼지 종류)) 연결한다. 반면에 클라이언트의 요청에 따라 해당 웹 사이트만 연결하는 크롤링 시스템에서는 요청할 때마다 1 번 연결한다. 따라서 클라이언트가 시간당 34 번 이상의 요청을 할 경우엔 자동화된 크롤링 시스템이 효율적이다.

② 클라이언트의 요청에서부터 서버의 응답까지 걸린 시간

자동화된 크롤링 시스템일 경우 클라이언트의 요청과 별개로 모든 데이터 정보들을 서버에 미리 준비해 놓는다. 따라서 데이터의 수집, 정제의 단계가 생략되므로 다른 방식에 비해 서버의 응답까지의 시간이 짧다. 2.2.2 에서 두가지 방법에 대해서 서버로부터 응답 받는 데까지 걸린 시간을 측정해본 결과(그림 11)는 자동화된 크롤링 시스템에선 4.17ms, 요청에 따라 동작하는 크롤링 시스템은 317ms 로 자동화된 크롤링의 응답시간이 짧다는 것을 알 수 있다.

따라서 ①, ② 두 가지 방면 모두에서 자동화된 크롤링이 효울적인 것으로 예상된다.

4. 결론 및 토론

본 연구에서는 크롤링 기법을 이용하여 실시간으로 미세먼지 정보를 제공해주는 모바일 어플리케이션을 구현하였다. 두 가지 방식의 크롤링 시스템을 구현해보았고 어떤 방식이 더 효율적인지 비교해보는 과정에서 필요한만큼만 크롤링 하는 방식이 언뜻 효율적으로 보였으나 크롤링할 정보의 특성상 그렇지 않았다. 앱 구성에 필요한 데이터 개수나 크롤링 해야할 페이지가 훨씬 방대하고 많은 양이라면 필요한 만큼만 크롤링 하는 방식이 더 효율적일 수도 있으나 한시간에 한번 만 업데이트할 필요가 있고 많지 않은 정보량을 갖는 데이터 특성상 자동화된 크롤링 방식이 더 효율적이었다.

데이터베이스를 연동하고 앱을 구성할 때 로그인 기능을 추가하거나 다른 유용한 기능들을 추가한다면 훨씬 더 완성도 있는 앱이 될 것이라고 생각하고 더 완성도 높은 어플리케이션을 만들지 못한 것이 아쉬웠지만 본 연구를 진행하면서 흔히 말하는 벡엔드(서버)와 프론트엔드(모바일 앱) 양쪽 모두의 개발 과정의 전반적인 흐름을 알 수 있게 되었고 크롤링 기법의 다양한 활용성에 대해서도 많은 것을 생각할 수 있었다.

참고문헌 – References

1. 박은선, 오현정, 김수현, 민아리. 대학생의 미세먼지 위험에 대한 인식, 지식, 관리행위에 대한 지각된 장애와 건강관리행위의 관계. 한국기초간호학회, 2018; 20(1):20-29
2. Wikipedia, Web crawler, https://ko.wikipedia.org/wiki/웹_크롤러/
3. Wikipedia, REST, https://ko.wikipedia.org/wiki/REST/
4. Redhat, https://www.redhat.com/ko/topics/api/what-is-a-rest-api
5. IBM korea, https://www.ibm.com/kr-ko/cloud/learn/rest-apis
6. Wikipedia, JSON, https://ko.wikipedia.org/wiki/JSON
7. JustKode.kr, https://justkode.kr/python/flask-restapi-1