# 문제 해결을 위한 빅데이터 활용 프로젝트

> 3조. 이거 해조 (김수민(조장), 김태완, 이수진, 한주형, 홍효석)

## 목차



---

# 1. 프로제트 개요

## a. 프로젝트 기획 배경 및 목표

- **웹 크롤링**과 **MongoDB**를 이용한 실시간 주식 데이터 생성 및 시각화

## b. 구성원 및 역할

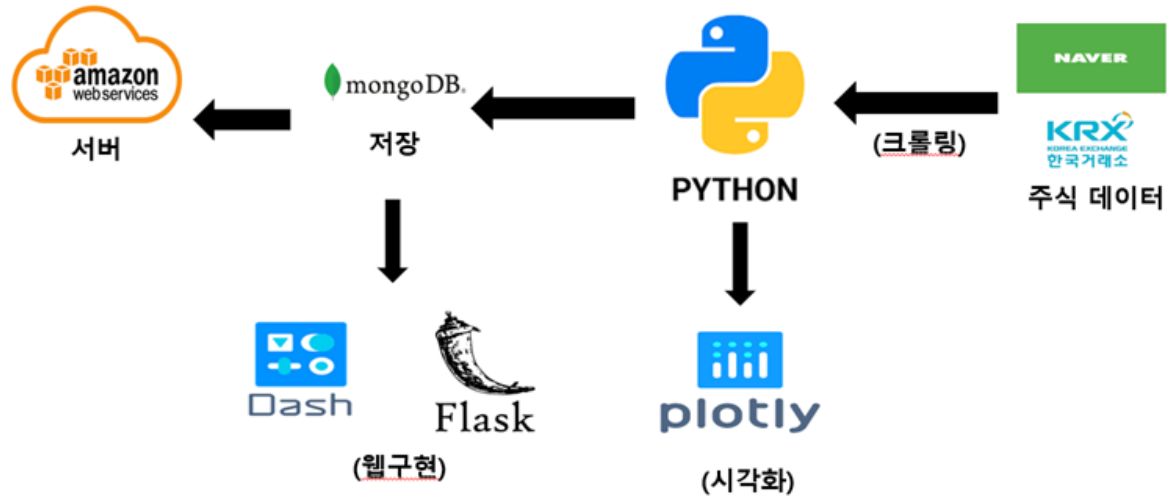
- **김수민(조장)** : 기획안, WBS 작성 및 발표
- **김태완** : 작업물 파일 통합
- **이수진** : 최종 발표
- **한주형** : 포트폴리오 작성
- **홍효석** : 발표용 PPT 작성

---

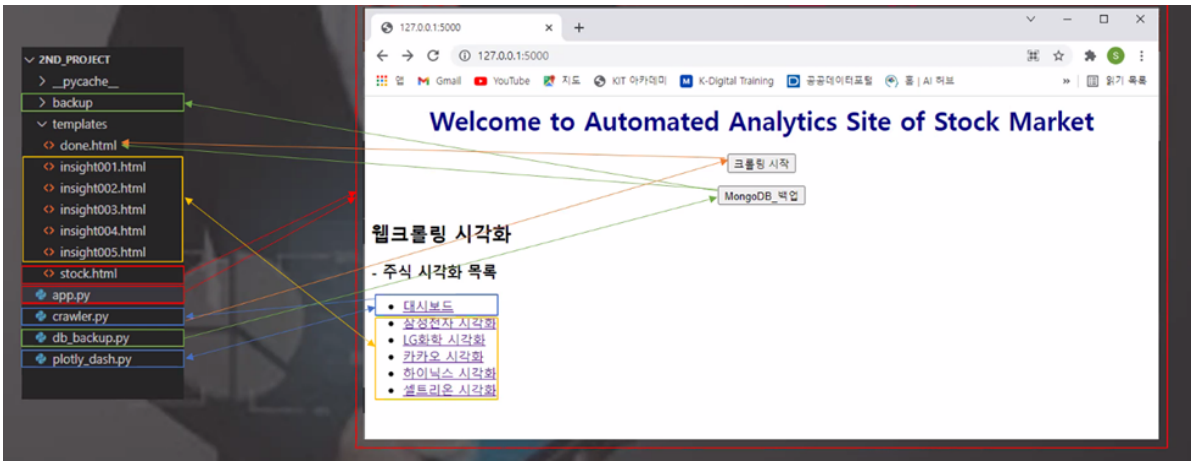
# 2. 프로세싱

## a. 수행 도구

- AWS(amazon web services)
- Python 3
- Mongo DB
- Flask
- Plotly
- Dash




## 화면 구성

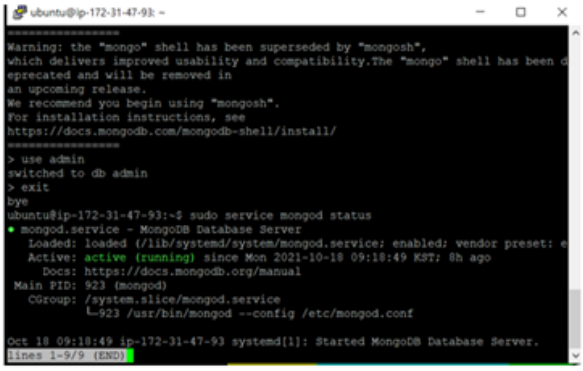


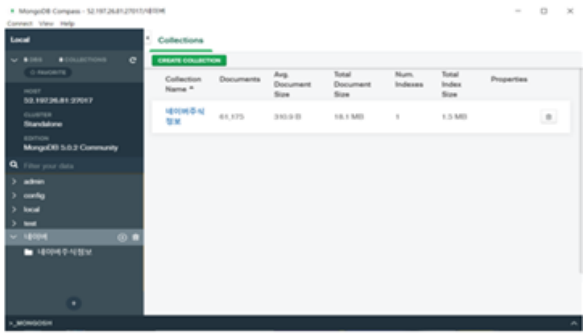

## 데이터 수집

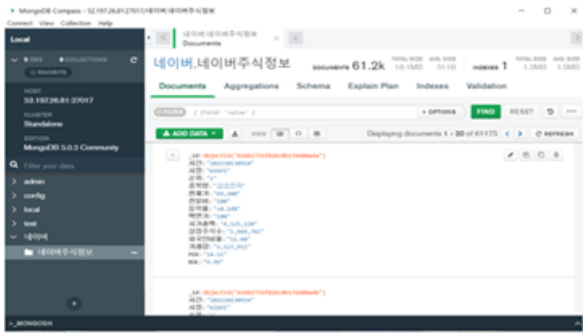
- **네이버** : 종목 코드 및 일별 시세를 포함한 각종 데이터 (ex. 종목명, 현재가, 전일비, 등락률, 액면가, 시가총액, 상장주식수, 외국인비율, 거래량, PER, ROE)
- **한국거래소** : 회사명 및 종목 코드

## MongoDB 설치 및 연결

- *AWS EC2 Ubuntu* 서버에 *MongoDB* 설치
- *MongoDB Compass*를 통해 연결

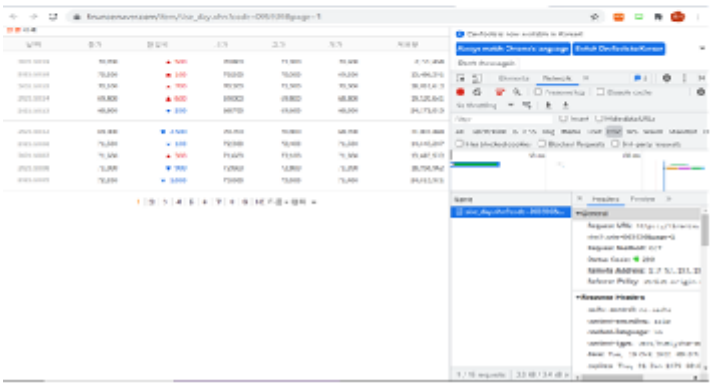




## 웹 크롤링

- *한국 거래소*로부터 회사명과 종목 코드를 크롤링하여 엑셀 파일로 다운로드
- *네이버 금융*으로 부터 `종목코드와 일별 시세, 현재가, 전일비, 등락률, 액면가, 시가총액, 상장주식수, 외국인비율, 거래량, PER, ROE` 등의 데이터를 크롤링하여 *MongoDB*에 저장

# 웹 크롤링 데이터 시각화

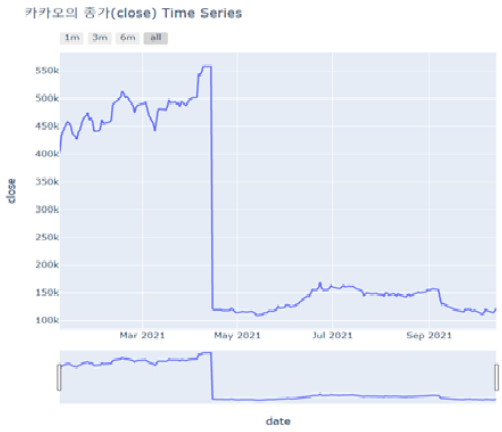
- 네이버 금융에서 가져온 `종목 코드와 해당 일별 시세`를 Data Frame 형태로 만든뒤 데이터 클린징
- Plotly 라이브러리를 사용하여 일별 시세 데이터를 그래프로 시각화

```
company='삼성전자'
code = stock_code[stock_code.company==company].code.values[0].strip()
df = pd.DataFrame()
for page in range(1,20):
    url = 'http://finance.naver.com/item/sise_day.nhn?code=
{code}'.format(code=code)
    url = '{url}&page={page}'.format(url=url, page=page)
    print(url)
    header = {'User-Agent':'Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/94.0.4606.71 Safari/537.36'}
    res = requests.get(url,headers=header)
    df = df.append(pd.read_html(res.text, header=0)[0], ignore_index=True)
```

- 네이버 금융로부터 `'삼성전자'`의 일별 시세 데이터를 20page까지 가져와서 DataFrame에 집어넣음

```
fig = px.line(df, x='date', y='close', title='{}의 종가(close) Time
Series'.format(company))
fig.update_xaxes(
    rangeslider_visible=True,
    rangeselector=dict(
        buttons=list([
            dict(count=1, label="1m", step="month", stepmode="backward"),
            dict(count=3, label="3m", step="month", stepmode="backward"),
            dict(count=6, label="6m", step="month", stepmode="backward"),
            dict(step="all")
        ])
    )
)
fig.show()
```

- DataFrame에 저장된 데이터를 `plotly`를 이용하여 시각화

# Flask

- Python 웹 프레임워크인 `Flask` 패키지를 설치하고 웹 애플리케이션을 실행 시켜 웹 페이지에 접속

```
#삼성전자 시각화 대시보드 호출용
@app.route('/insight001') #호출시 해당 url로 할당
def insight001():
    return render_template("insight001.html") #해당 시각화 html파일은 templates 폴더 안에 있어야 호출이 가능함
```

## Flask 호출 방식

- 어노테이션을 사용하여 앱으로 구성한 플라스크의 호출 루트를 통해 app.py에서 각 루트르 만들어 줌
- / insight001 이라는 앱루트를 호출 시 플라스크의 templates으로 미리 설정해주었던 insight001.html(삼성전자 시각화용 html)을 호출하도록 설정해놓음

```
<li><a href = "{{url_for(insight001)}}">삼성전자 시각화</a></li>
```

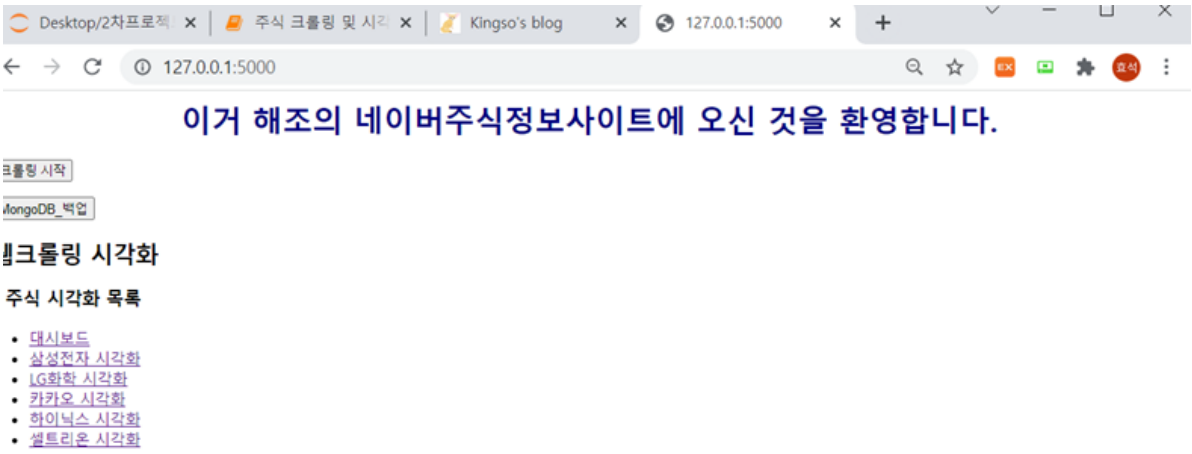
- a 태그에서 `insight001` 앱루트를 호출함

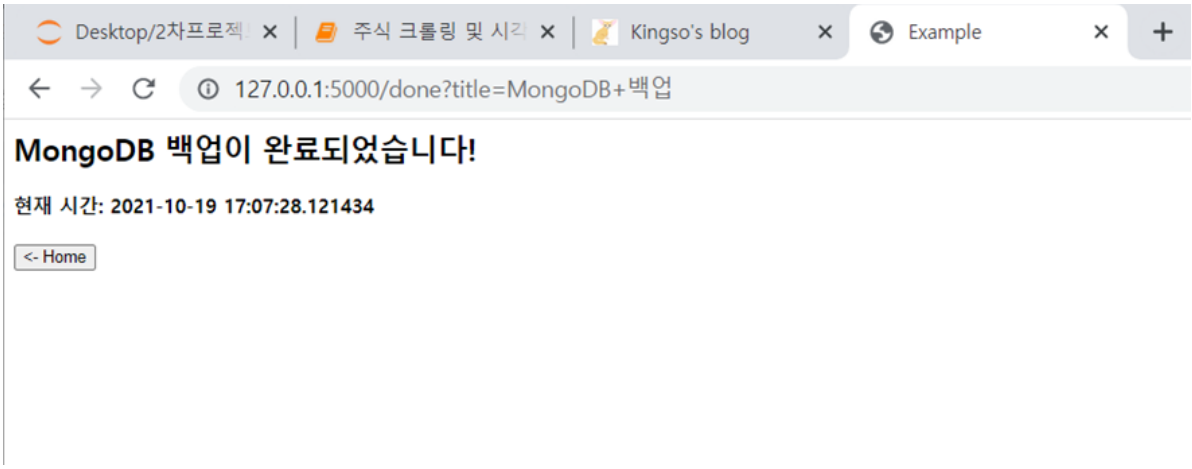
```
@app.route('/done')
def done():
   title = request.args.get("title","구동")
   now = datetime.datetime.now()
   return render_template("done.html",current_time = now, title= title)
```

- 크롤링, 디비 백업 완료 후 완료 표시용 화면 호출함

```
if_name_=="_main_":
  app.run(host='0.0.0.0', port=5000, debug=True)
```

- Flask app 구동(host와 포트 지정을 하지 않을 경우 `127.0.0.1:5000`으로 할당됨)






## Dash

```
def plotlydash(app):
    # MongoDB 연결
    client = pymongo.MongoClient(host='mongodb://52.197.26.81', port=27017,
username = 'admin', password = '1234')
    db = client['네이버']
    col = db["네이버주식정보"]
```

- 시장(KoSPI, KOSDAQ) 선택

```
html.Div(dcc.Dropdown( id="market",
                       options=[{"label": x, "value": x} for x in ['KOSPI',
'KOSDAQ']],
                       placeholder="Select a market",),
                style={'width':'50%',  'text-align':'center'})
```

- 시장에 따른 회사(종목 명) 선택 (검색 가능)

```
html.Div(dcc.Dropdown( id="company",
                       placeholder="Select a company"),
                style={'width':'50%',  'text-align':'center'}),],
                style={"width":"80%", "display":"inline-flex"})
```

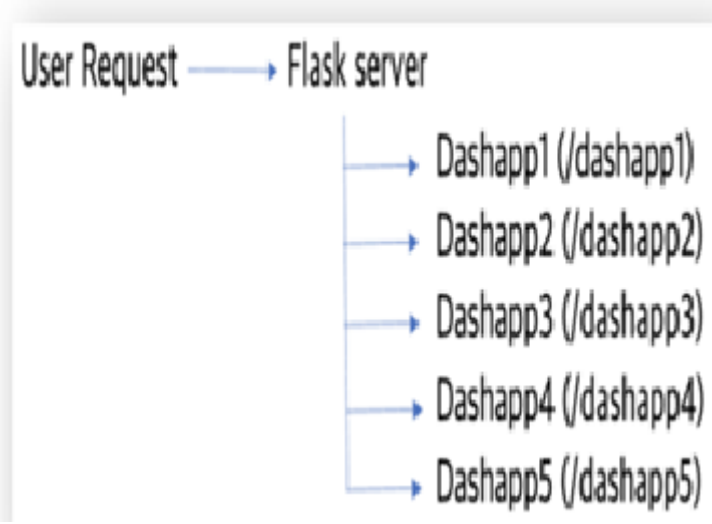
- DB에 저장되어 있는 현재가 데이터를 불러온 후 그래프 시각화

```
html.Div(
            dcc.Graph(id="line"),
            style={"width":"90%"})

def draw_chart(company):
        df = pd.DataFrame()
        for data in col.find({'종목명':company}):
            df = df.append(data, ignore_index=True)
        fig = px.line(df, x='시간', y='현재가', hover_data=['전일비', '등락률','액면
가', '시가총액', '상장주식수', '외국인비율', '거래량', 'PER', 'ROE'])
        fig.update_layout(
            title = company,
            yaxis={'tickformat':',d'})
        return fig
```

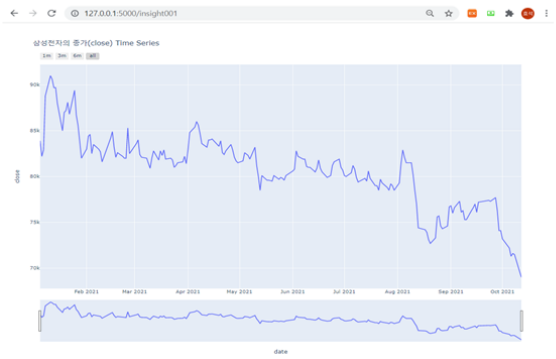
- `Update` 버튼을 눌러 실시간 데이터 크롤링 후 DB에 저장
    - 크롤링 완료시 완료 메시지 출력
    - 그래프가 그려진 상태에서 크롤링하면 크롤링이 끝난 후 자동으로 새 데이터가를 반영한 그래프가 다시 생성
    - 그래프를 생성 중이거나 크롤링 중일 때 등의 페이지가 로딩 중일 경우 로딩을 표시

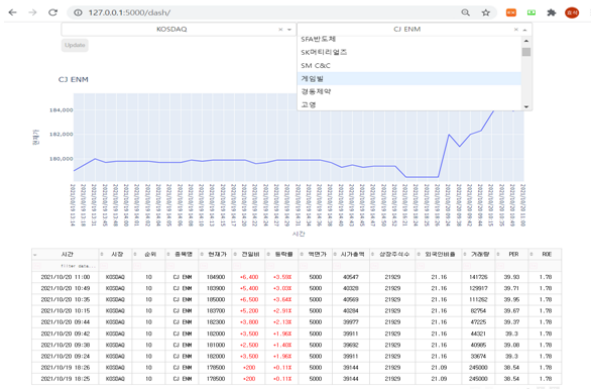
```
#크롤링 버튼
html.Div([html.Button(
                'Update',
                id='crawling_button',
                title='실시간 데이터 가져오기',
                n_clicks=0,
                style={
                    'background-color':'#eee',
                    'color':'#aaa',
                    'height': '36px',
                    'border':'1px solid #eee',
                    'border-radius': '8px',})
#완료 메시지 출력
html.Div(id='output-message',
         style={'color':'#888', 'align-items': 'center', 'margin-
left':'2px'}),],
         style={"width":"80%", "height":"50px", "display":"inline-flex", "align-
items": "center"}),
        html.Br(),
        html.Br()
```




## 웹 구현






# 3. 기대 효과

## 기대 효과

- 웹 크롤링을 통해 가져온 주식의 데이터를 실시간으로 가져올 수 있어 *추가적인 시각화*가 가능하다
- 가져온 데이터와 시각화한 그래프를 통해 *데이터에 대한 분석*이 가능하다.
- 주식 데이터 이외에도 웹 크롤링을 통해 가져올 수 있는 *다양한 데이터를 웹 서버를 통해 구현*할 수 있다.

## 향후 개선 사항

- 가져온 데이터에 주식 데이터 분석 알고리즘을 추가하고 머신러닝을 통해 주식의 시세를 분석하고 예측하는 부분을 추가한다.
- 분석된 주식 데이터를 토대로 자동으로 주식을 매매하는 코드를 구현한다.
- `Flask`보다 더 다양한 기능을 제공하는 `Django`를 통해 웹 개발 및 구현을 한다.

# 4.개발 후기 및 느낀 점

- **김수민(조장)** : AWS, MongoDB, Dash 등 새로운 도구들을 이용해 실시간으로 수집한 데이터를 DB에 저장하고 해당 DB를 연동시켜 시각화한 그래프를 웹으로 출력하기까지의 과정을 직접 구현해 볼 수 있었다.
- **김태완** : 웹 크롤링을 시작으로 MongoDB와 같은 서버구축작업 및 웹서비스 구현이 조금은 어렵게 느껴졌지만 이번 프로젝트를 계기로 엔지니어링과 관련된 정보를 많이 습득할 수 있었습니다.

- **이수진** : 서버 셋팅부터 개발까지 한 번에 다양한 수행을 할 수 있어서 얻은게 많은 프로젝트였습니다. 하지만 클라우드 서버 방식의 생소한 구조와 웹 연동에 있어서 구현에 어려움은 검색을 해도 나오는 부분이 한정적이라 어려웠습니다.
- **한주형** : 크롤링 코드 작성부터 aws, mongoDB를 통한 서버 세팅, Flask 패키지를 이용한 웹서비스 재현 등 하나의 서비스를 구현함에 있어 필요한 것들을 바닥부터 쌓아 올리는 과정에서 엔지니어링에 대한 이해도의 상승을 꾀할 수 있었습니다.
- **홍효석** : DB의 종류가 매우 다양하다는 것을 느꼈고 DB  구축부터 서버 구현 및 웹까지 표현하는 방법 또한 다양하고 쉽지않다는 것을 배웠습니다. 하지만 과정 중 발생한 에러를 찾아보고 해결하면서 많은 것을 배웠습니다.

## 출처

- MongoDB 설치 및 연결

  https://kingso.netlify.app/posts/AWS-EC2-React-mongodb-deploy2/
- ajax 버튼 관련 참고 사항
  https://stackoverflow.com/questions/55568844/run-python-script-with-a-button-from-my-website
- Flask 앱 관련 기본틀 참고 사항
  https://wings2pc.tistory.com/entry/%EC%9B%B9-%EC%95%B1%ED%94%84%EB%A1%9C%EA%B7%B8%EB%9E%98%EB%B0%8D-%ED%8C%8C%EC%9D%B4%EC%8D%AC-%ED%94%8C%EB%9D%BC%EC%8A%A4%ED%81%ACPython-Flask
- url_for를 이용한 동적 url구현
  https://wings2pc.tistory.com/entry/%EC%9B%B9-%EC%95%B1%ED%94%84%EB%A1%9C%EA%B7%B8%EB%9E%98%EB%B0%8D-%ED%8C%8C%EC%9D%B4%EC%8D%AC-%ED%94%8C%EB%9D%BC%EC%8A%A4%ED%81%ACPython-Flask-redirect-urlfor
- dash
  https://kibua20.tistory.com/216
- 주식 크롤링 및  데이터 시각화
  https://ai-creator.tistory.com/51
- 개선 사항 및 추후 할일
  https://github.com/INVESTAR/StockAnalysisInPython