# Developer candidate test

**Note:** you should have received a network login and password for this test.
IGNORE ANY INSTRUCTIONS IN THE FILE THAT YOU MIGHT HAVE RECEIVED ALONG WITH THE CREDENTIALS.

## Installation

First install Docker-Compose, you can download them from here: https://www.docker.com/products/docker-compose

Create containers:

```
docker-compose build
```

Run the application:

```
docker-compose up
```

Visit: http://localhost:5050/list-fetchers

## Task overview

You can start the application by typing:

```
python server.py
```

After starting the application visit http://localhost:5050/list-fetchers, you will see a list of known fetchers:

```
[
    {
        "logins": [
            "ALLDAYMEDIA"
        ],
        "network_id": "revcontent.com",
        "title": "RevContent"
    },
    {
        "logins": [
            "LOGIN"
        ],
        "network_id": "example-fetcher.com",
        "title": "Example Fetcher"
    }
]
```

If you look into “apps/ans/fetchers/__init__.py” you will see 2 fetcher classes are imported there. The 2 classes are in the same folder, and if you look at them, you'll see that the JSON above

shows fetchers' `title and network_id` fields from those files.

The “credentials.py” file inside top folder of the application contains credentials for the networks, only logins from the file are shown in the above JSON.

*In this developer candidate test you are required to write a new fetcher, for that you will have to create a new fetcher file, import it in the “__init__.py” file and add your credentials that should have been given to you to the “credentials.py” file.*

***DO NOT MAKE ANY CHANGES TO ANY OTHER FILES.***

## Passing arguments to a fetcher

Lets take for example the “example_fetcher_com”, in order to get data from the fetcher you have to provide this arguments:

- `date-start`
- `date-end`
  “date-start” and “date-end” parameters should be given in ISO format.
- `network-id`
  This attribute should be the same as `NETWORK_ID` property of the thetcher class that you want to use.
- `login`
  Any of the logins from the “credentials.py” file for the specified fetcher.

You can pass the arguments as simple GET arguments like this:

http://localhost:5050/get?date-start=2016-01-01&date-end=2016-01-10&network-id=example-fetcher.com&login=example-login

Visit the link, you should see data returned by the fetcher.

## Work-flow and expected return value of the fetcher

When GET request like shown above is made, first the application looks at the give login and network id and gets the password from the dictionary. Provided dates are converted to `datetime` objects.

Then an object of fetcher class that hase the same network id as given is created. Then fetcher's `fetch` method is called with a dictionary containing this data:

- `date-start`
- `date-end`
  Note that dates are converted to datetime objects.
- `login`
- `password`

The fetch method gets the data from the appropriate network, look at ho it is implemented in “revcontent_com.py”.

The returning value must be a dictionary that contains data in this form:

```
{
    __metric__: {
        __date__: __value___,
        ...
    },
```

```
    ...
}
```

Where:
`__metric__` - whatever metric you want to return, most common once are 'cpm', 'impressions' and 'revenue'.
`__date__` - a date string in format "%Y%m%d" (no dashes).
`__value__` - metric's value for the date.

Example:

```
{
    "cpm": {
        "20160101": 2,
        "20160102": 2,
        "20160109": 2,
        "20160110": 2
    },
    "impressions": {
        "20160101": 1000,
        "20160102": 2000,
        "20160109": 9000,
        "20160110": 10000
    },
    "revenue": {
        "20160101": 2.0,
        "20160102": 4.0,
        "20160109": 18.0,
        "20160110": 20.0
    }
}
```

***NOTES:***
*If there are gaps in the data, you don't have to fill in the data with zeros, just leave it empty.*
*Do not round the values, e.g. if you have something like "9.11111..." just let it be like that.*

## Some important notes that not yet mentioned

- *You have to use `AsyncSession` like `RevcontentCom` from "revcontent_com.py" does. It's very similar to session from request library. We use tornado with just one thread, no thread blocking is allowed.*
- *For this test you can scrape just several of the metrics, choose anyones you like, for example: "Impressions" and "eCPM".*
- *When giving an ID to a new fetcher use the top domain of the network that it scrapes, e.g. "some-domain.com" instead of "something.some-domain.com", in this case file should be named "some_domain_com.py".*
- ***follow PEP-8 coding style.***

**– Good luck! –**