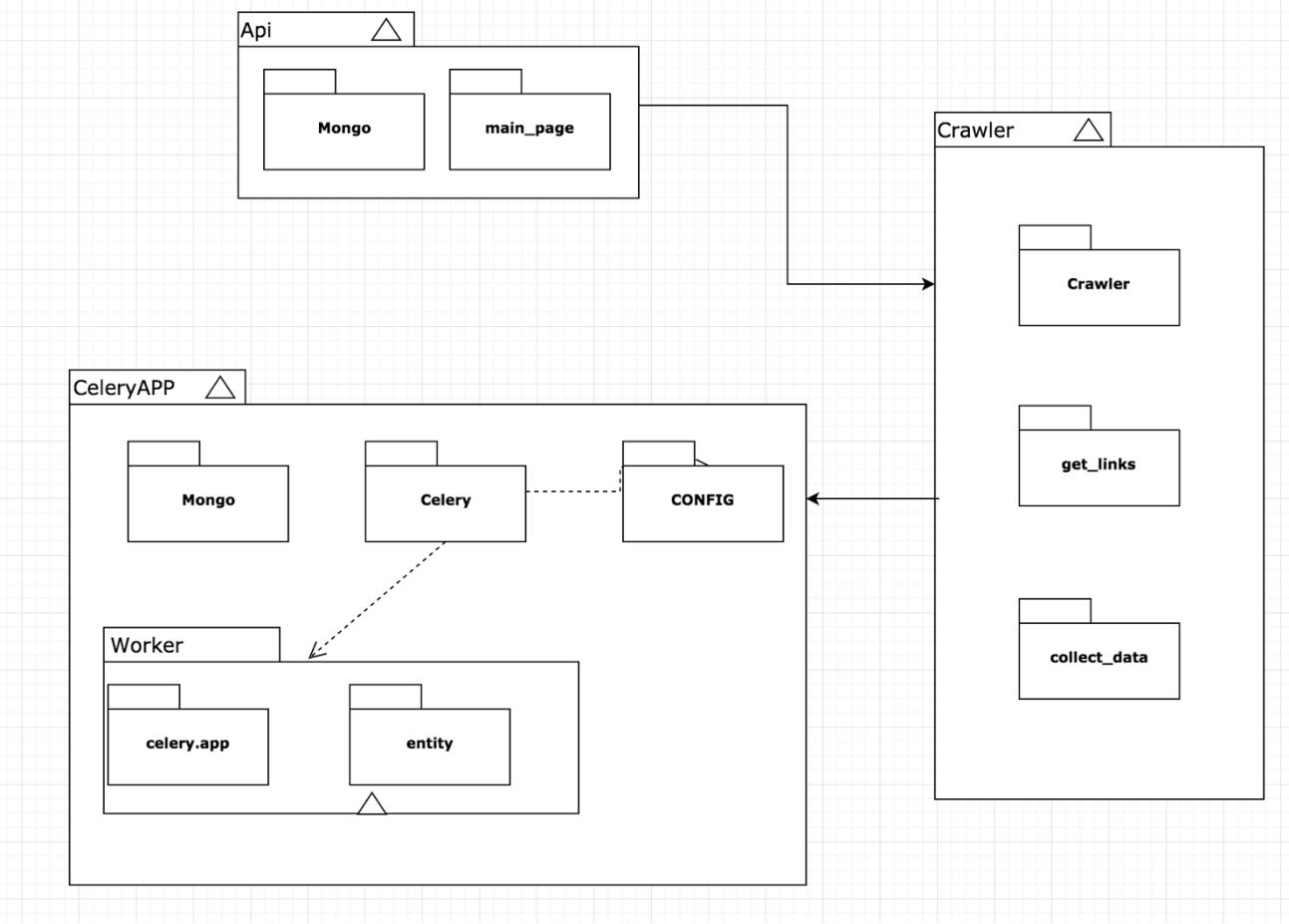
26 January 2018

TRIPY: CRAWLER FOR TRIPADVISOR

According to our task, the main goal is collecting data from well-known site for travelers – TripAdvisor.



1. REQUIMENTS

This crawler is based on the following techniques:

- MongoDB
- RabbitMQ
- Python 3.6 with next packages:
    a. Celery – Distributed Task Queue
    b. PyMongo – driver for MongoDB
    c. Requests – HTTP library
    d. lxml – paring HTML actually
    e. fake_useragent – for overcoming the 301 response from server

2. INSTALLATION

[Master] MongoDB

First of all, we need a Database for storing our crawled data. Taking into account the hierarchical and quite complex structure of data, we have selected *MondoDB* as a database for our purpose.

*If you already have a configured MongoDB on your server, feel free to skip this part of guide.

1. Install MongoDB on your server;
2. Create a user with ***readWrite*** role and access to the ***TripY*** db;

```
db.createUser({user: 'exam', pwd: 'A', roles: [{role: 'readWrite', db: 'TripY'}]})
```

3. Enable auth and remote access to MongoDB

```
a) bindIp: 0.0.0.0  <- change this line
b) security:
       authorization: 'enabled'
```

4. Open 27017 port for remote connection to your db: `sudo ufw allow 27017`
5. Restart MongoDB: `sudo service mongod restart`

We have finished with database installation now.

[Master] RabbitMQ

For making easier next step, we've prepared some bash scripts, which you can find in project repo. However, if you prefer manual installation – it's your choice!

*If you already have a configured RabbitMQ on your server, feel free to skip this part of guide.

1. Install *git* to your server;
2. Clone project repo:

```
git clone https://github.com/AlexWorldD/TripY/
```

3. Make RabbitMQ installation script executable: `chmod +x MakeRabbit.sh`

4. Run: `./MakeRabbit.sh`

After all, you should see the final message like:

```
                                                                    [OK]
        RabbitMQ web management console
        URL: 159.65.17.172:15672
```

We've finished with required stuff on the [Master] side, so now we can prepare our workers.

[Worker] Docker and Docker-compose

Have a look on this tutorial.

[Client] Install all packages from *requiments.txt*

3. CONFIGURATION

Cloning repo from git

First of all, we should clone project repo to our [Worker] machines.

```
git clone https://github.com/AlexWorldD/TripY/
```

Configure MongoDB and RabbitMQ addresses

Change IPs in config file (/ TripY/cluster_managment/default_config.py)

```
CELERY_BROKER_URL = 'amqp://<user>:<pwd>@<IP>:<Port>'
MONGO = 'mongodb://<user>:<pwd>@<IP>/<DataBase>'
```

For instance, our configuration was:

```
CELERY_BROKER_URL = 'amqp://radmin:a@159.65.17.172:5672'
MONGO = 'mongodb://exam:A@159.65.17.172/TripY'
```

Additionally, in this file you can select the option for crawling reviews: True or False. However, you should take into account the fact that common entity has roughly 500 reviews, what could make your crawling process critical slow.

Run Dockers

```
docker-compose up --force-recreate --build --scale worker=4
```

The last parameter sets the number of workers, we strongly recommended set it according to *1 GB per worker*, otherwise some problems could be happened.

4. USAGE

Our crawler just works, and it's amazing!

Run *main.py* and type the city name (or first letters of city name), if it's real city which is already in TripAdvisor database.

Wait until the crawl will finished the parsing links to entities and broadcast them to workers. After that, you can *off* your client. And connect via your preferable client to MongoDB with given address and User/Pwd pair.

5. CRAWLED DATA

Hotel/Attraction/Restaurant

- Title
- Address
- Link
- Rating
- Prices
- Contacts: official site and phone number – optionally
- Specific features
- Reviews – optionally

Review

- Entity ID
- UserID
- User nickname
- Date
- Title
- Full text
- Ratings

All Data in database has crossing field (such as GEO_ID, ID or User_ID), what gives the opportunity easily getting specific data after crawling process.