RSS Feed Scraper

Web application to monitor RSS Feed

by Abhilash Ajay

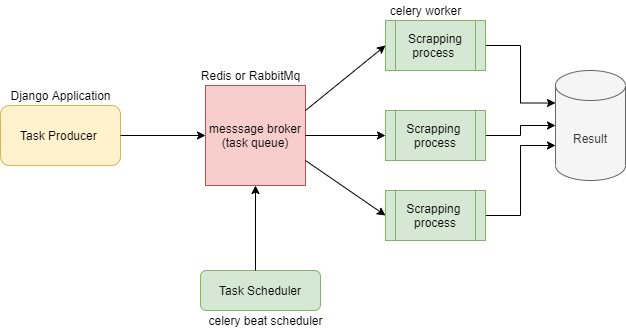
Objective:

* Provide RSS Feed to the user based on subscription
* Display the top 10 Feed subscribed by the users
* User should get access of new notification which contains title, a summary of the text content
* Look for news article in the feed every hour

Software tools:

* DJANGO for client Application
* Celery for task scheduler and worker
* Reddis or rabbitmq as message broker

Diagram of the proposed System:



Working of the system:

1. The Scrapping Process are assigned as tasks in the system

* The Summary generation tools used are feedparser / newspaper3k / (requests using BeautifulSoup and JSON)

Link for the scraping script made by requests: <https://github.com/abhilashajay-dev/rss_scraping/blob/main/scraping.py>

1. Django application which hosts the models for the project and saves the result of the worker process in the db

* News\_feed model which uses User as its foreignkey
* Where User is an instance of the django’s built in User

1. Task scheduler is done by celery which is intergrated in Django \_\_init\_\_.py in main source file and celery.py in the same directory as the settings.py of the django application

Code inside celery.py:

from \_\_future\_\_ import absolute\_import, unicode\_literals

import os

from celery import Celery

os.environ.setdefault('DJANGO\_SETTINGS\_MODULE','<PROJECT NAME>.settings')

app = Celery('pro')

app.config\_from\_object('django.conf:settings', namespace='CELERY')

app.autodiscover\_tasks()

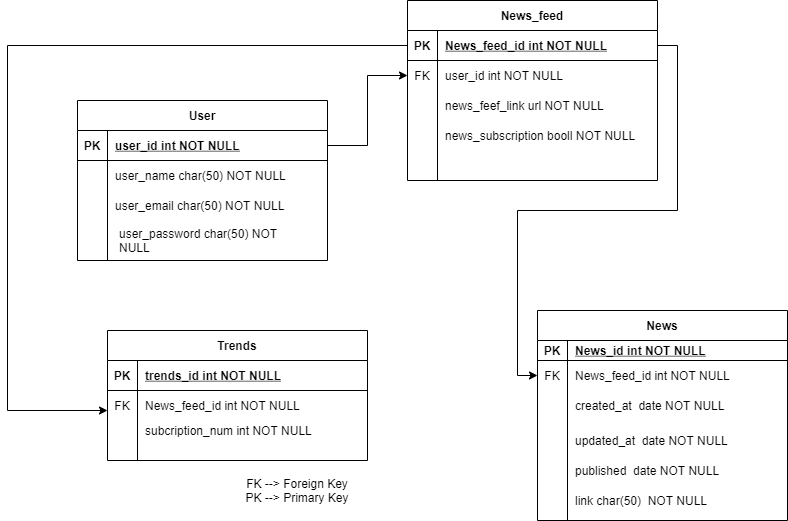
1. Message broker contains task queue for the celery worker

Where it is distributed to each worker process.

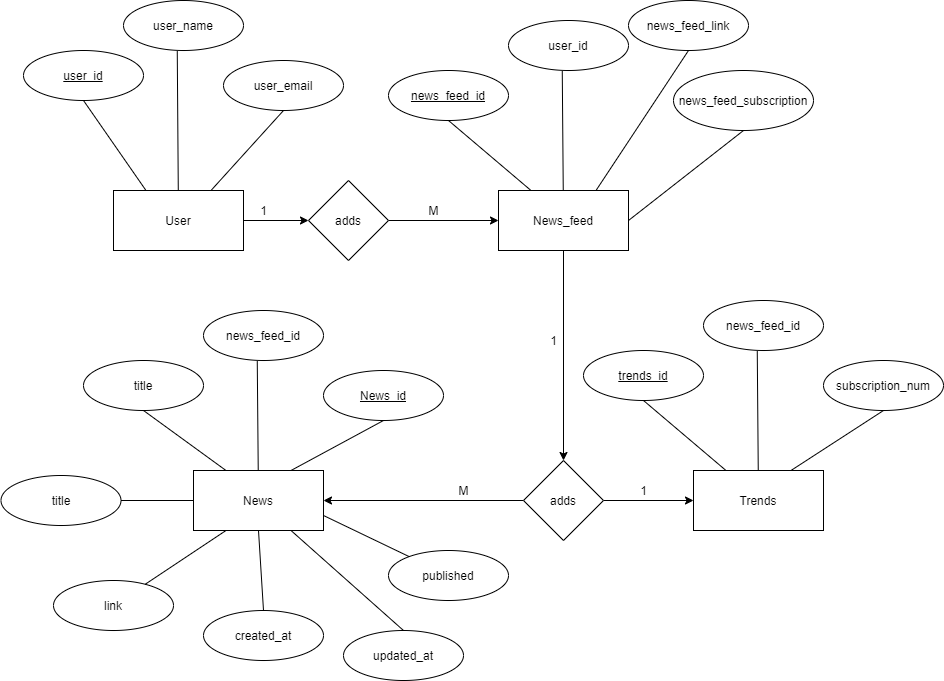
This is done by either redis or rabbitmq

Note: redis need to be configured in the settings.py

Table Design:



ER Diagram:



Components involved int the application:

* Django application first develop a model for storing input data which could be a form with a link of RSS feed the User want to scrape into the newsfeed model
* Taskmanager Next integrate celery into the project by adding celery.py in the main src folder with the django.settings
* Message-broker Config the django setting to point to redis/rabbitmq server

Eg:

Settings.py

# celery

CELERY\_BROKER\_URL = 'redis://localhost:6379' # redis server address

CELERY\_RESULT\_BACKEND = 'redis://localhost:6379'

CELERY\_ACCEPT\_CONTENT = ['application/json']

CELERY\_TASK\_SERIALIZER = 'json'

CELERY\_RESULT\_SERIALIZER = 'json'

CELERY\_TIMEZONE = 'UTC'

* Scraper Component Create a scraping app and add to the django project

Inside the scraping app create tasks.py -🡪 this will contain the code used for scrapping import the News model to create a new entry everytime the script gets activated which can be setup by using celery.crontab or perodic scheduler

* Subscription check on Newsfeed objects System is setup in such way that only the subscribed Newsfeeds get go to the broker messager to extract RSS Feed from the link given in the object NewsFeed.
* Updation of trends displays most subscribed RSS feed Each time similar Newsfeed is subscribed(create/update News\_feed) by a User Trends gets a signal to update the subscription sum by checking if the object exists and update the subscription\_sum+=1

Link to the scaled down version of my rss\_scraping project is attached below:

django\_rss\_scrapping:

<https://github.com/abhilashajay-dev/rss_scraping/tree/main/django_rss_scraping>

my github repo:

<https://github.com/abhilashajay-dev>