Web crawler for Economic Development Organizations in the US

Summary

High-level design of a web crawler that finds contact data for Economic Development Organizations (EDOs) in the US, organized by cities.

Author: Jesus Sanchez
Created: July 2024
Last updated: July 2024

Glossary

- **EDO**: Economic Development Organization
- **IRS**: Internal Revenue Service
- **API**: Application Programming Interface; intended to interact with the web crawler results
- **MVP:** Minimum Viable Product
- **The project**: referred as the whole application (Web crawler + API)

Detailed Design

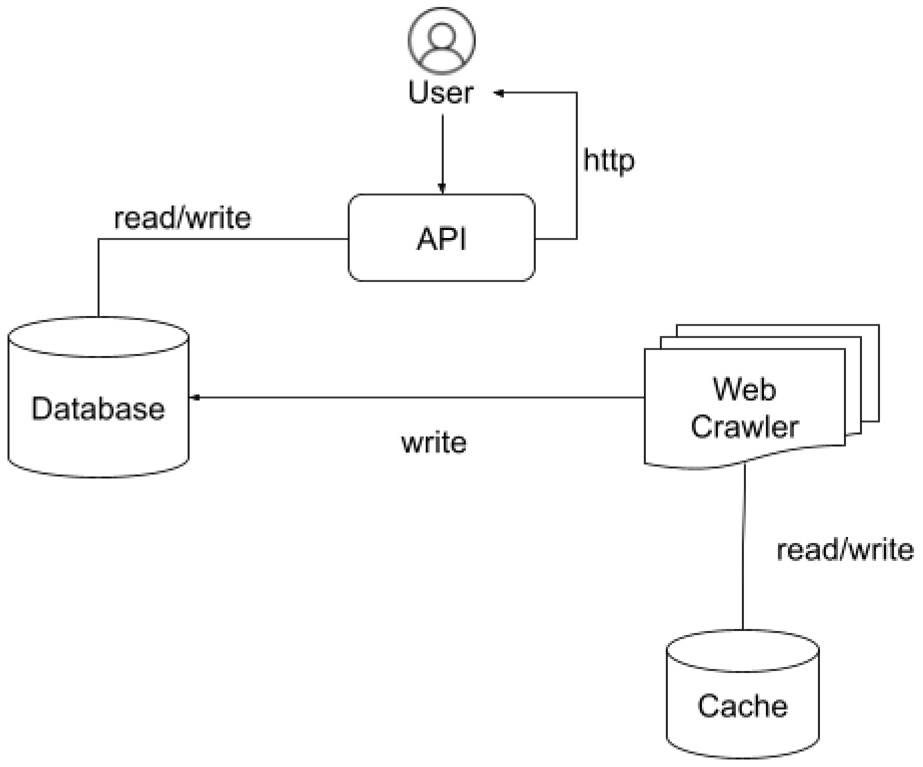
The project consists of 4 main stateless microservices divided into
- Web crawler
- Cache
- Database
- API

Tech Stack

- **Docker** is used for the deployment and administration of the microservices. Docker is widely used in the industry for its containerization capabilities, making it a good choice for both MVP projects and highly scalable applications.
- **Python** is the primary programming language for the web crawler and the API. Python has useful libraries developed by the community that facilitate data analysis.

- **PostgreSQL** is the database that stores the results of the web crawler. A relational database is helpful for normalizing data, although a non-relational database could also be considered for projects that do not require this.
- **RedisDB** is used as the cache due to its simplicity and suitability as a key-value store.

Architecture



Web Crawler

The web crawler starts automatically each time the application is deployed, initially using the default city "New York". After this first run, users can execute the script on demand for different US cities. The process follows these steps:

- **Retrieve Seed Data**: Fetch initial data from the IRS website. For example, you can access data for New York from the URL: https://www.irs.gov/pub/irs-soi/eo_ny.csv. Unfortunately, the IRS data may not include email addresses and websites.
- **Search on Cause IQ**: Use Google to search for the EDO name followed by "causeiq". Cause IQ is a website that provides information on nonprofit organizations, such as phone number and website
- **Google search** missing data from the previous step (phone and website) and pick the first result (Not always accurate results are returned for this step)

- **Email Address**: For the email address, rely on a search query such as "<EDO name> email".

Note: Between each individual step, we leverage caching to avoid hitting rate limiting and throttling issues while scraping the web.

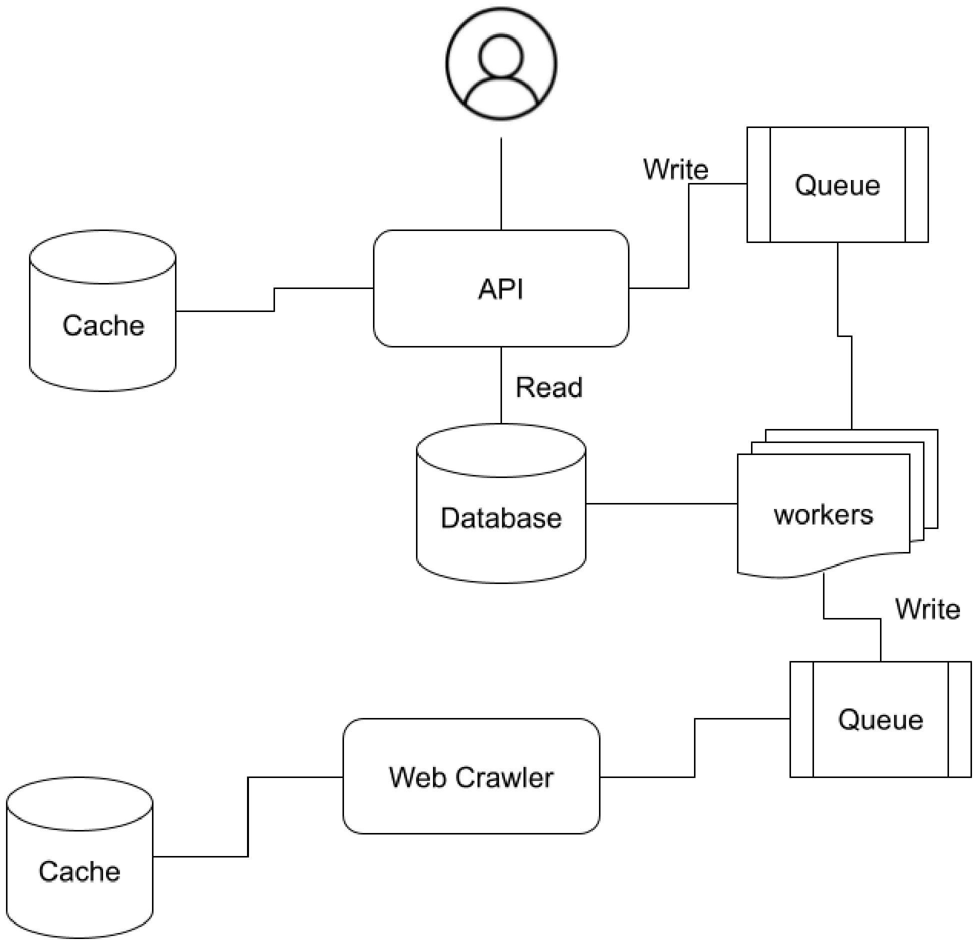
Non-Goals

The project is not intended for a production environment, although it is a stepping stone towards that goal. The following issues should be addressed first:

- **Usage of New Microservices**: Implement microservices as workers dedicated to I/O connections with the database.
- **Queue System**: Implement a queue system for communication between microservices.
- **Web Crawler Optimization**: The web crawler currently retrieves information from multiple instances and uses caching between those processes. A better system design would involve breaking this into multiple services connected through a queue system to reduce cache consumption.
- **Data Encoding Strategy**: Currently, the app encodes pandas dataframes using the standard Python library pickle. While pandas is excellent for data analysis, its main purpose is not for data storage. Consider using a more scalable data encoding method, such as protobuf.
- **Automated Tests**: Due to time constraints, automated testing was not initially incorporated into the project. However, it is essential to implement unit and integration tests between each project step.

Proposed Architecture

For a production environment the following architecture is suggested


Write
Queue
Cache
API
Read
Database
workers
Write
Queue
Web Crawler
Cache