Career Portal

Architectural Diagram is attached at the bottom

Microservices:

Account Service
Candidate Service
Job Service
Notification Service
Email Service
Job Search Service
Recruiter Search Service

Databases:

Cassandra for Account Service:

We have decided to use Cassandra as our preferred database for to handle account service, because we believed achieving availability, and Partition Tolerance is a priority also the fact that Cassandra is tunable, it provides us with option to reconfigure and in achieving consistency as needed. Hence, making our database highly scalable

MongoDB for Notification Service:

We decided to use MongoDB as our preferred database for Notification service because we believe consistency and partition-tolerance is a priority in this case as we expect lots of data that are not directly related with each other. Hence, we are able to leverage partition-tolerance and consistency providing us with easily scalable database.

MySQL for Candidate Service:

To store applicant's information, we have used a MySQL database. We came to a decision to use MySQL for different reasons. The first this is that a job-seeker entity has a relationship with many other entities such as skills, experience, education and certificates. In order to be able to map this relationships, we believe a relational database would be more appropriate. The other and most important benefit we saw as a reason for our decision to use MySQL database for the candidate micro-service is that we believe applicant's profile should be highly available and consistent. Since there number of applicant's profiles only grows with the number of applicants, for the number of user we have at the beginning, MySQL will be able to handle the amount of data and partition may not be necessary. Hence, we can leverage the benefits of using relational database such as normalization and optimization as well as maintain data that's not redundant.

Communication between micro-services:

The candidate service uses a product service to send message to the kafka topic we created for the appropriate topic. In the candidate service, we have created a topic called ‘candidate modified topic’. Hence, the candidate service can send a job-seeker object as a string to the candidate modified topic which is also mapped as a string in a Kafka template our producer service manages. In this way, any service interested in the applicant’s information will be able to consume.

Other Third-Party Services:

Kafka:

Our Kafka implementation contains three topics – Candidate Modified, Job Created, and email topics. The candidate services send message asynchronously to the Candidate Modified topic, the message being the job-seeker object at creation. The Job service sends message asynchronously the message being a new vacancy to the Job Created topics. The Notification service sends message asynchronously the message being the email of the applicant to the email topic. The consumers in our Kafka implementation are the email service and the notification service which are internal services that don’t expose an end point to the public network. The notification service consumes the job-seeker object and store it in a database and also consumes a vacancy to detect new postings. The email service then consumes the email and sends new job postings alerts to registered job-seekers.

Elastic Search:

Since most of the requests are going to be search requests for a job, to be able to provide fast and robust search service for Job search service and Recruiter search service, we have decided to use elastic search service. That way we can avoid hitting the data base for each request instead leverage elastic search services ability to store information for fast search.

Deployment:
GCP is used for deployment, and we have created a cluster. Inside the cluster we have multiple replicas of multiple services and multiple pods are running to cater our project.
==========

GCP Career Portal Structural Flow

Daniel Gebretsadik | Ala Farah | Usama Karam | Ahmed Abdeen | Mohammad Khader | Safwan Abbasi | Usman Haider

June 17, 2022