Abdiganiyeva Uldana, 190103171, FX Software Engenering

Assume that you have to build search engine

- 1 Describe its design.

- 2 Describe users of the system

- 3 What kind of architecture you will use

- 4 What kind of stack of technologies you will use to build it

- 5 How would you test your system

1. Suppose we need to build a search engine based on online ticket sales. Users entering the site enter information about their trip or flight into the search engine. Our system will process and produce results based on their requests. When a user enters a query. It is first processed by a specialized device that automatically redirects the user's request to the least loaded cluster at the moment. This makes the most efficient use of the available computing power.

The search request then goes to a metasearch engine. This system retrieves all the necessary data and finds out what type of data the request belongs to. At the same stage, the request is checked for spelling.

Then the user's request is passed on to the "basic search" servers. It is in the basic search is the index of the search engine, divided into separate parts and distributed across servers, as the search by parts is always faster.

It is worth noting that each server has multiple copies. This allows not only to protect the information from loss, but also to distribute the load. If the information from a particular server turns out to be too much in demand and one of the servers is overloaded, the problem is solved by connecting copies of that server.

Based on the search results, each basic search server returns to the metasearch server the results associated with the user's request. Next, a sorting algorithm is connected to work, which will give the results to the user given his selected filters. (Cheapest, fastest)

2. To search for tickets we will use robots that will process the request (by date, by location, by price, by mode of transport).I count them as users within the system.

What types of robots will we have?

Global. Collect data on all modes of transport;

mirrors. Identify resource mirrors;

checkers. Monitor the presence of the resource in the search engine database and the number of indexed documents;

researchers. Used for debugging search algorithms and studying individual sites;

fast robots. Automatically check the date of the last update and promptly index new information.

Well, the main users will be people who want to buy tickets, plan vacations, etc.

3. In the first phase for projects such as search engines, I think the focus is on quality requirements, and the functional requirements may be weaker. A high degree of parallelism, a large amount of memory and a fast query are the lifeblood of a search engine.

I believe that a client-server architecture model would be suitable for our search engine. The users will come to the site and make queries in the search engine. After that the information will be processed in the controller in two steps which will give the results according to the MVC principle. The first step is the scanning and analysis module, which regularly scans and analyzes our database. Save the scanned and analyzed information in the database. For the database, it is basically divided into two parts: a links table to store link structures, a content table to store web page content, a table of ticket data(number, price, description, where from, where to, date, flight time, etc.). Other parts of the database that are not related to the search engine will not be considered. Let's assume that they are already there. The search engine is only a subsystem of the big ticketing system.

The second step is to process the user's request and return the search result to the user. Since the database of up to several servers is redundant, you only need to search on that server when searching. The search module accepts the user's request and performs word segmentation and synonym processing on the user's request. It then searches the result set by querying the index table and the content table, and the result set is weighted and sorted according to the user's specified requirements.

The next stage of the user interface runs on another web server. The web server presents the web page and accepts the user's request. First, it parses the user's

input for sensitive words, sends the parsed request to the background server, retrieves the search results from the background server, and passes the results through View to the user.

The business logic will be located in the model.

With this architecture, the system can be expanded, adding new items or search criteria will not affect its performance. A stable Internet connection will provide fast connection to the database and update information. If a large number of users visit the site at the same time, there will be no problems.

4. Technology stack. We will use React to build the site. For database storage we will use Apache, PostgreSQL. Back to Java Spring. When the client will book a ticket, using the "book" button, we reserve and fill in a seat and remove this ticket from the data where free tickets will be located. If in that case the client returns the tickets, we reset the data and add it to the database again.

5. System testing

1) I, as a user, go to the site, click on the search box. I enter keywords and click on the search button. The site gives me tickets that match my requests. The search engine is working correctly.

2) I, as a user, specify filters when searching, such as preferred airlines, only seats near the window, talgo or reserved seat, etc. The information on my request corresponds to my requests.

3) I, as a user, have booked a ticket that I want to purchase. When searching for the same ticket as I booked, there was no result. The booked ticket cannot be found at the time of booking. The system is working correctly