Frameworks used

1. **Load balancer**: This piece of software is generally responsible for distributing the load among multiple endpoints, without we hardcoding the endpoints. For example if we have 2 endpoints as [http://localhost:8080](http://localhost:8080/) and [http://localhost:8081,](http://localhost:8081,/) without a load balancer, we would have to hardcode this list somewhere. With a load balancer in place we wouldn’t have to do that.**Ribbon**, is one of the famous Spring security solution for load balancing.
2. A **network discoverer**: For a load balancer to automatically route all its requests to endpoints, we would want a network discovery software. The network discoverer would usually register any new instance coming up. The load balancer can then contact the network discoverer to find out the available endpoints and route its requests based on a round robin mechanism. **Eureka** is the most preferred network discoverer and Ribbon is designed to work with it seamlessly, when used with Feign.
3. **API Gateway**: API Gateway is a critical piece of software required to route requests inside our microservices network. An API Gateway like **zuul**, sits on top of Eureka, where it registers itself and is available for all the requests. An API Gateway can also be used for advanced features like Rate-limiting, shared sessions or security as well.
4. **Spring** Cloud **Sleuth** is meant to help with this exact problem. It introduces unique IDs to your logging which are consistent between microservice calls which makes it possible to find how a single request travels from one microservice to the next.
5. **Feign (OpenFeign) :** It is a framework that allows easy creation of REST clients. Feign's first goal was reducing the complexity of binding Denominator uniformly to http apis regardless of restfulness. It uses **Ribbon** under the hood.
6. **Hystrix**:  If a microservice is down or not functioning properly then the issue may cascade up to the upstream services. Netflix created Hystrix library implementing the [Circuit Breaker pattern](https://martinfowler.com/bliki/CircuitBreaker.html) to address these kinds of issues. We can use **Spring Cloud Netflix Hystrix Circuit Breaker** to protect microservices from cascading failures.
7. **Swagger :** for API documentation. <http://localhost:8000/swagger-ui.html>

**About the Project :**

A traveller can travel from one city to another via any mode eg: AIR, ROAD, TRAIN

Current database consists of 3 tables:  
  
**City** :   
id, name  
We other parameter as related to city like address, type, etc  
  
**Line**:  
id,line\_name,line\_mode,ORIGIN\_ID,DESTINATION\_ID,departure\_time,arrival\_time  
here line\_mode is mode travel : Air / Road / Train etc.  
  
**Route** : If there are connecting city or stop in the line route  
id,LINE\_ID,stop\_number,CITY\_ID,arrival\_time,departure\_time  
Here stop\_number is the sequence number of the halt

After running the Crud-service which will create and connect to the H2 in-memory database. We have to execute the below queries for inserting the data in database.  
SQL file is available in /resource folder

INSERT INTO city(id,city\_name) values(1,'Mumbai');

INSERT INTO city(id,city\_name) values(2,'Delhi');

INSERT INTO city(id,city\_name) values(3,'Nagpur');

INSERT INTO city(id,city\_name) VALUES (4,'Hyderabad');

INSERT INTO city(id,city\_name) VALUES (5,'Agra');

INSERT INTO city(id,city\_name) VALUES (6,'Pune');

INSERT INTO city(id,city\_name) VALUES (7,'Chennai');

INSERT INTO city(id,city\_name) VALUES (8,'Jaipur');

INSERT INTO line(id,line\_name,line\_mode,ORIGIN\_ID,DESTINATION\_ID,departure\_time,arrival\_time)

values(1,'Air1','AIR',1,2,'01:00:00','02:15:00');

INSERT INTO line(id,line\_name,line\_mode,ORIGIN\_ID,DESTINATION\_ID,departure\_time,arrival\_time)

values(2,'Air2','AIR',1,4,'02:00:00','03:30:00');

INSERT INTO line(id,line\_name,line\_mode,ORIGIN\_ID,DESTINATION\_ID,departure\_time,arrival\_time)

values(3,'Bus1','ROAD',1,6,'02:00:00','06:30:00');

INSERT INTO line(id,line\_name,line\_mode,ORIGIN\_ID,DESTINATION\_ID,departure\_time,arrival\_time)

values(4,'Train1','TRAIN',6,2,'00:30:00','23:30:00');

INSERT INTO line(id,line\_name,line\_mode,ORIGIN\_ID,DESTINATION\_ID,departure\_time,arrival\_time)

values(5,'Train2','TRAIN',4,5,'00:30:00','23:30:00');

INSERT INTO route(id,LINE\_ID,stop\_number,CITY\_ID,arrival\_time,departure\_time)

values(1,4,1,1,'03:30:00','03:40:00');

INSERT INTO route(id,LINE\_ID,stop\_number,CITY\_ID,arrival\_time,departure\_time)

values(2,4,2,3,'07:30:00','07:35:00');

INSERT INTO route(id,LINE\_ID,stop\_number,CITY\_ID,arrival\_time,departure\_time)

values(3,4,3,8,'16:00:00','16:35:00');

INSERT INTO route(id,LINE\_ID,stop\_number,CITY\_ID,arrival\_time,departure\_time)

values(4,3,1,3,'04:00:00','04:05:00');

INSERT INTO route(id,LINE\_ID,stop\_number,CITY\_ID,arrival\_time,departure\_time)

values(5,5,1,1,'04:00:00','04:05:00');

INSERT INTO route(id,LINE\_ID,stop\_number,CITY\_ID,arrival\_time,departure\_time)

values(6,5,2,3,'04:00:00','04:05:00');

INSERT INTO route(id,LINE\_ID,stop\_number,CITY\_ID,arrival\_time,departure\_time)

values(7,5,3,8,'04:00:00','04:05:00');

SELECT \* FROM ROUTE ;

SELECT \* FROM LINE ;

SELECT \* FROM CITY ;

**Netflix-Eureka-Naming-Server :** network discovery service

**Crud-Service** : to manipulate resources this service is used. Currently, if we pass a City name, it will return all the travel lines whose start point is this city or whose route is connected with this city.

GET : Localhost:8000/all/{cityName}  
This can also be accessed via Zuul api gateway  
http://localhost:8765/crud-service/all/{cityName}

**Search-Service** : if we pass the start and the end city, this service will return the itinerary.  
GET [http://localhost:8100/itinerary/from/{from}/to/{to}](http://localhost:8100/itinerary/from/%7bfrom%7d/to/%7bto%7d)

**Netflix-Zuul-Api-Gateway-Server** : Api Gateway.

**Spring-Cloud-Config-Server**: Cloud config server. To maintain all the configuration at one centralised location.

Search-Service

Crud-Service

Get : City List

H2

Netflix-Zuul-Api-Gateway-Server

Netflix-Eureka-Naming-Server

Config-Server