Urban clap Backend System

Developed By : **Vatsal Hirpara**

Microservice Identification:

I have identified the following microservices according to functionalities or sub domain.

CUSTOMER : data and APIs related to customers

WORKER : data and APIs related to workers (service providers)

SERVICE : data and APIs related to services provided to customers(e.g. haircut, bath fitting)

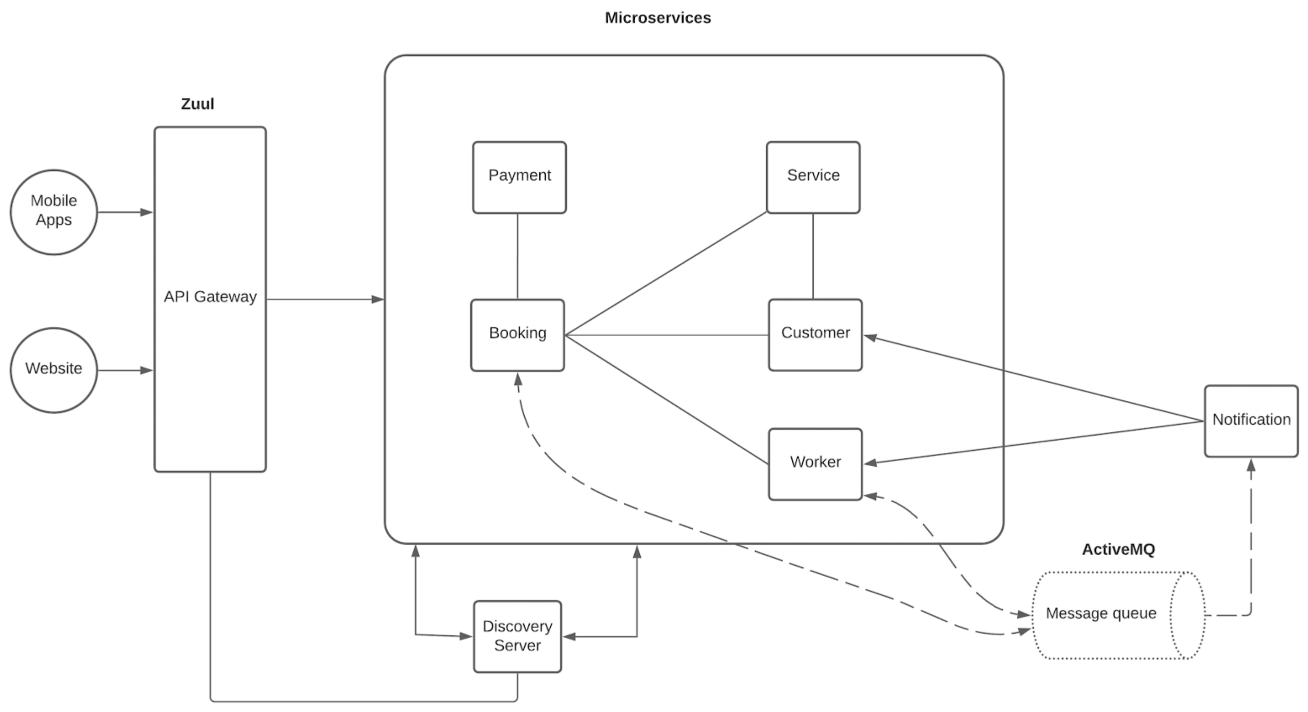
BOOKING : data and APIs related to mapping of service, customer and assigned worker

PAYMENT : for now by default it returns successful payment, but in real case scenario, payment related apis can be implemented in this microservice

NOTIFICATION : contains endpoints to send and receive notifications for customers and workers.

I have created separate microservices for **API gateway** and **Discovery server** .

High level diagram :



REST endpoints exposed and application flow:

API gateway is running on port 9999 and Eureka server is running on port 8761.

1) Get list of services available

Microservice : **SERVICE**

Path : http://localhost:9999/services

Method : GET

Response :

```
{
  "error": false,
  "errors": null,
  "data": [
    {
      "id": 1,
      "serviceCategory": "SALON_FOR_MEN",
      "serviceName": "HAIRCUT",
      "price": 299.0
    },
    {
      "id": 2,
      "serviceCategory": "APPLIANCE_REPAIR",
      "serviceName": "AC_REPAIR",
      "price": 189.0
    },
    {
      "id": 3,
      "serviceCategory": "PLUMBERS",
      "serviceName": "BATH_FITTING",
      "price": 449.0
    }
  ]
}
```

2) Get list of customers

Microservice : **CUSTOMER**

Path : http://localhost:9999/customers

Method : GET

Response :

```
{
  "error": false,
  "errors": null,
  "data": [
    {
      "id": 1,
      "name": "Amit Saxena",
      "email": "amit@gmail.com",
      "city": "Gurugram",
      "address": "OM, sector 18, opp mall, gugrugram"
    },
    {
      "id": 2,
      "name": "Karan Patel",
      "email": "Raj@gmail.com",
      "city": "Noida",
      "address": "Pretige heights, sector 56, Noida"
    },
    {
      "id": 3,
      "name": "Rohit Sharma",
      "email": "Rohit@gmail.com",
      "city": "Delhi",
      "address": "Lotus panache, sector 17A, Delhi"
    }
  ]
}
```

Select one of the service ids and select one of the customer ids. Now send a booking request for a customer and a service . Let's select customer 1 and service1(HAIRCUT) for example.

Note : You might get a gateway timeout error here for sending a POST request to booking for the first time due to some issue in docker networking(works fine outside docker). Please send the same POST request (below step 3)once again, it will work fine.

3) Book a service request.

   Microservice : **BOOKING**
   Path : http://localhost:9999/bookings/book-service
   Method : POST
   Request :

```
{
    "customerId": 1,
    "serviceId": 1
}
```

Response :

```
{
  "error": false,
  "errors": null,
  "data": {
    "id": "d6ed0dfc-ef80-456d-9a54-eaf524a8b211",
    "bookingStatus": "PROCESSING",
    "customerId": 1,
    "serviceId": 1,
    "serviceCategory": "SALON_FOR_MEN",
    "serviceName": "HAIRCUT",
    "workerId": null,
    "workerName": null,
    "creationTimeStamp": "2021-02-28T15:48:29.743772Z"
  }
}
```

After sending a booking request, the status of booking is **PROCESSING** and booking id **d6ed0dfc-ef80-456d-9a54-eaf524a8b211** . Since worker is not assigned yet workerId and workerName is null right now.
Now after a brief moment a worker will be assigned and bookingStatus will change to **WORKER_ASSIGNED.**

To fetch the booking details, send a request to the following API.

4) Get Booking by booking id
Microservice : **BOOKING**
Path : http://localhost:9999/bookings/{bookingId}
Method : GET
Request : GET http://localhost:9999/bookings/d6ed0dfc-ef80-456d-9a54-eaf524a8b211
Response:

```
{
  "error": false,
  "errors": null,
  "data": {
    "id": "d6ed0dfc-ef80-456d-9a54-eaf524a8b211",
    "bookingStatus": "WORKER_ASSIGNED",
    "customerId": 1,
    "serviceId": 1,
    "serviceCategory": "SALON_FOR_MEN",
    "serviceName": "HAIRCUT",
    "workerId": 1,
    "workerName": "Ramesh",
    "creationTimeStamp": "2021-02-28T15:48:29.743772Z"
  }
}
```

Now as you can see, status bookingStatus is changed to **WORKER_ASSIGNED** and worker with id 1 is assigned to perform service of haircut. A worker will receive notification regarding service assigned.

Now let's check the notification of the worker with id 1.

5) Check notifications of worker by workerId

Microservice : **NOTIFICATION**

Path : http://localhost:9999/notifications/workers/{workerId}

Method : GET

Response:

```
{
  "error": false,
  "errors": null,
  "data": [
    {
      "userId": 1,
      "text": "You have received service request for HAIRCUT and booking id :
d6ed0dfc-ef80-456d-9a54-eaf524a8b211"
    }
  ]
}
```

Now a worker can either accept or deny a booking request. To accept booking, send request to http://localhost:9999/workers/acceptBooking/{bookingId} and and to reject booking send request to http://localhost:9999/workers/rejectBooking/{bookingId}

Now let's decide to accept booking for worker (id:1) and booking id : **d6ed0dfc-ef80-456d-9a54-eaf524a8b211**

6) Accept booking by bookingId

Microservice : **WORKER**

Path : http://localhost:9999/workers/acceptBooking/{bookingId}

Method : GET

Example :

GET http://localhost:9999/workers/acceptBooking/d6ed0dfc-ef80-456d-9a54-eaf524a8b211

Response:

```
{
  "error": false,
  "errors": null,
  "data": "service accepted by worker"
}
```

Now service has been accepted by worker 1. Now customers will get notification regarding details of the worker and worker will get details of the customer.

Let's check the notification of customer 1.

7) Check notifications of customer by customerId

Microservice : **NOTIFICATION**
Path : http://localhost:9999/notifications/customers/{customerId}
Method : GET
GET http://localhost:9999/notifications/customers/1
Response:

```
{
  "error": false,
  "errors": null,
  "data": [
    {
      "userId": 1,
      "text": "Your booking of HAIRCUT with booking id
d6ed0dfc-ef80-456d-9a54-eaf524a8b211 has been Accepted by our service provider Ramesh
(email : ramesh@uc.com)"
    }
  ]
}
```

8) Let's check notification of worker 1
GET http://localhost:9999/notifications/workers/1
Response :

```
{
  "error": false,
  "errors": null,
  "data": [
    {
      "userId": 1,
      "text": "You have received service request for HAIRCUT and booking id :
d6ed0dfc-ef80-456d-9a54-eaf524a8b211"
    },
    {
      "userId": 1,
      "text": "Your booking has been scheduled for customer : Amit Saxena
Email(amit@gmail.com) for service HAIRCUT"
    }
  ]
}
```

see 2nd notification of worker for customer details.
If worker would have rejected then customer would get following notification :

```
{
      "userId": 1,
      "text": "Your booking of HAIRCUT with booking id d6ed0dfc-ef80-456d-9a54-eaf524a8b211
has been rejected by our service provider Ramesh (email : ramesh@uc.com)You will be assigned
another worker soon"
}
```

Interservice Communication Approach :

For synchronous communication, RestTemplate is used.

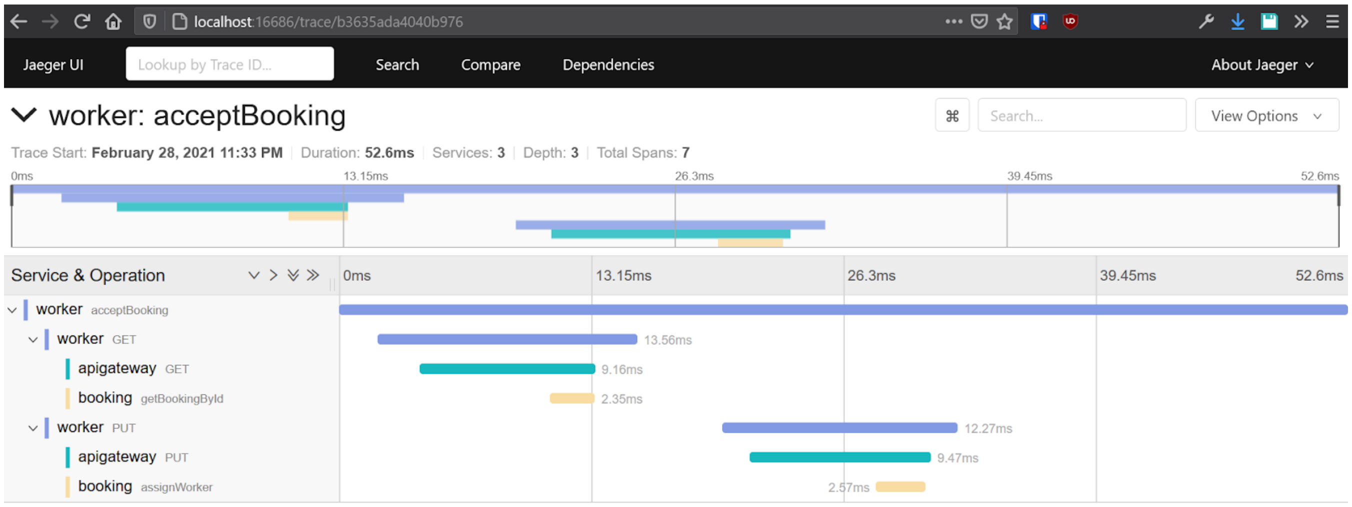
Since the number of microservices are comparatively less and according to my design there is no need for multiple consumers for a single event, I have used **ActiveMQ** as a message broker for async communications. Every event is only consumed by one microservice.

Following are details of event produced and respective consumer details:

Event	**Producer Microservice**	**Consumer Microservice**	**Description**
BookingRequestReceived	**BOOKING**	**WORKER**	Created when a booking request is received and worker microservice consumes it and assigns worker according to requirement
AcceptOrRejectBooking	**WORKER**	**NOTIFICATION**	After assigning a worker, worker microservice emits event and notification microservice consumes it and sends notification to worker regarding booking details.
BookingAcceptedByWorker	**WORKER**	**NOTIFICATION**	If a worker accepts booking, this event is produced and notification microservice sends notification to customer regarding worker details. Also sends notification to worker regarding customer details for booking.
BookingRejectedByWorker	**WORKER**	**NOTIFICATION**	If a worker rejects booking, this event is produced and notification is sent to the customer.

Distributed Tracing Screenshots:

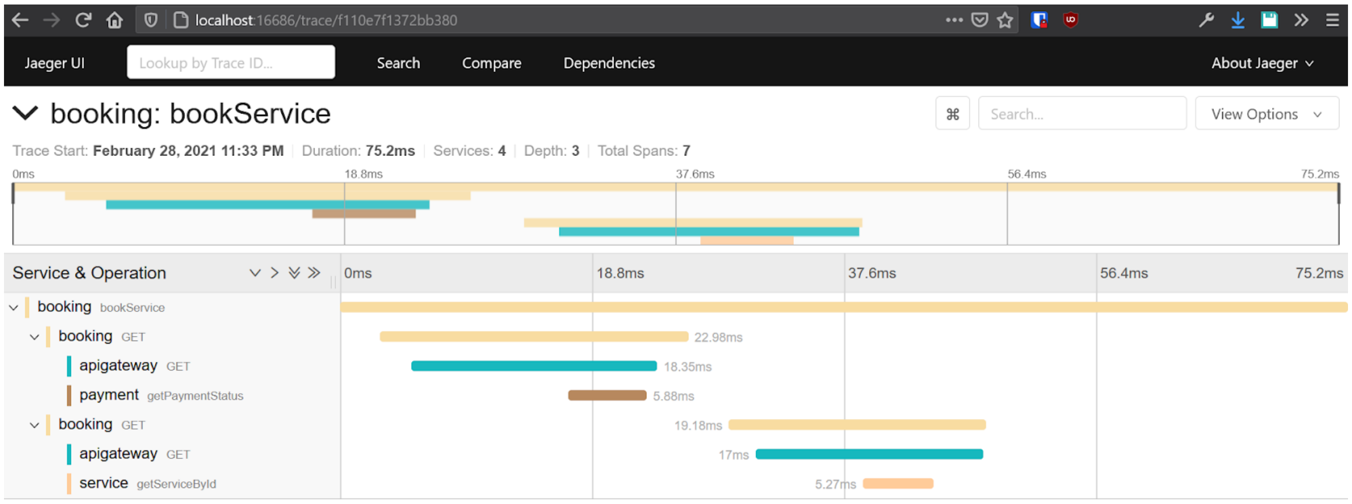
You can see distributed tracing at Jaeger dashboard at http://localhost:16686
username : admin, password: admin



worker: acceptBooking

Trace Start: **February 28, 2021 11:33 PM** | Duration: **52.6ms** | Services: **3** | Depth: **3** | Total Spans: **7**

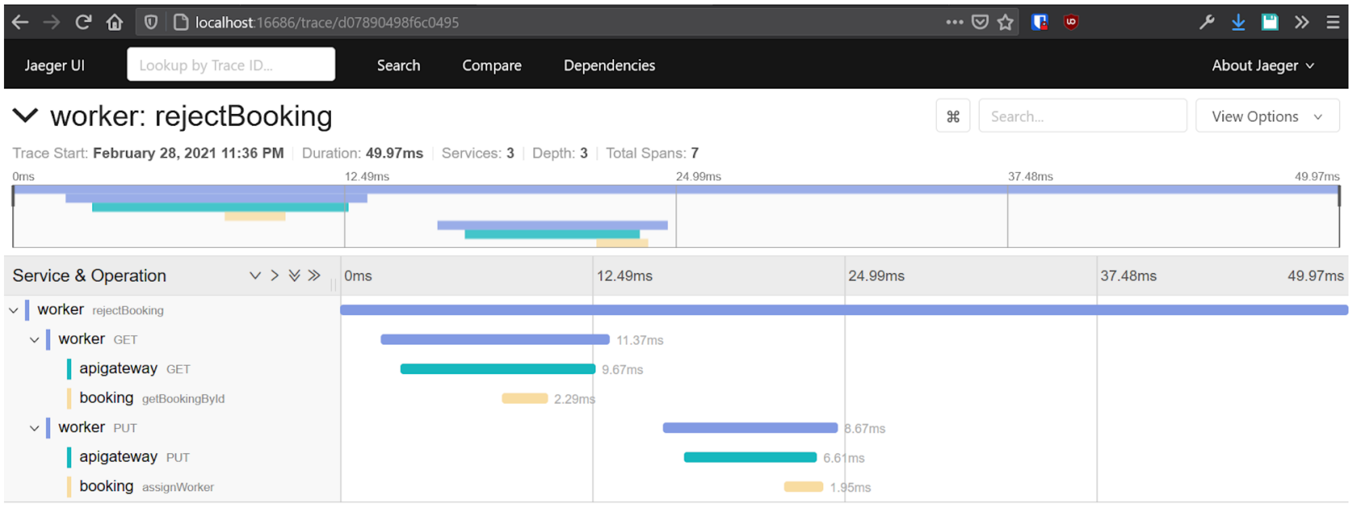
Service & Operation	Duration
worker acceptBooking	
worker GET	13.56ms
apigateway GET	9.16ms
booking getBookingById	2.35ms
worker PUT	12.27ms
apigateway PUT	9.47ms
booking assignWorker	2.57ms



booking: bookService

Trace Start: **February 28, 2021 11:33 PM** | Duration: **75.2ms** | Services: **4** | Depth: **3** | Total Spans: **7**

Service & Operation	Duration
booking bookService	
booking GET	22.98ms
apigateway GET	18.35ms
payment getPaymentStatus	5.88ms
booking GET	19.18ms
apigateway GET	17ms
service getServiceById	5.27ms



worker: rejectBooking

Trace Start: **February 28, 2021 11:36 PM** | Duration: **49.97ms** | Services: **3** | Depth: **3** | Total Spans: **7**

Service & Operation	Duration
worker rejectBooking	
worker GET	11.37ms
apigateway GET	9.67ms
booking getBookingById	2.29ms
worker PUT	8.67ms
apigateway PUT	6.61ms
booking assignWorker	1.95ms

Source code of microservices :

I have included source code with this zip. Here is the link to git repos of same code

CUSTOMER : https://github.com/VatsalHirpara/customer

WORKER : https://github.com/VatsalHirpara/worker

SERVICE : https://github.com/VatsalHirpara/service

BOOKING : https://github.com/VatsalHirpara/booking

PAYMENT : https://github.com/VatsalHirpara/payment

NOTIFICATION : https://github.com/VatsalHirpara/notification

API gateway : https://github.com/VatsalHirpara/apigateway

Discovery server : https://github.com/VatsalHirpara/discoveryserver

Docker-Compose-file: :
https://github.com/VatsalHirpara/urban-clap-deployment/blob/main/docker-compose.yml

Docker Hub link : https://hub.docker.com/u/vatsal09