Booking Assignment

Introduction

This assignment provide an examples for a microservices system. Your responsibilities is analyze the business requirement and basic design, translate it to detail design and programming it with Golang & GRPC.

There are some main technologies stack requires:

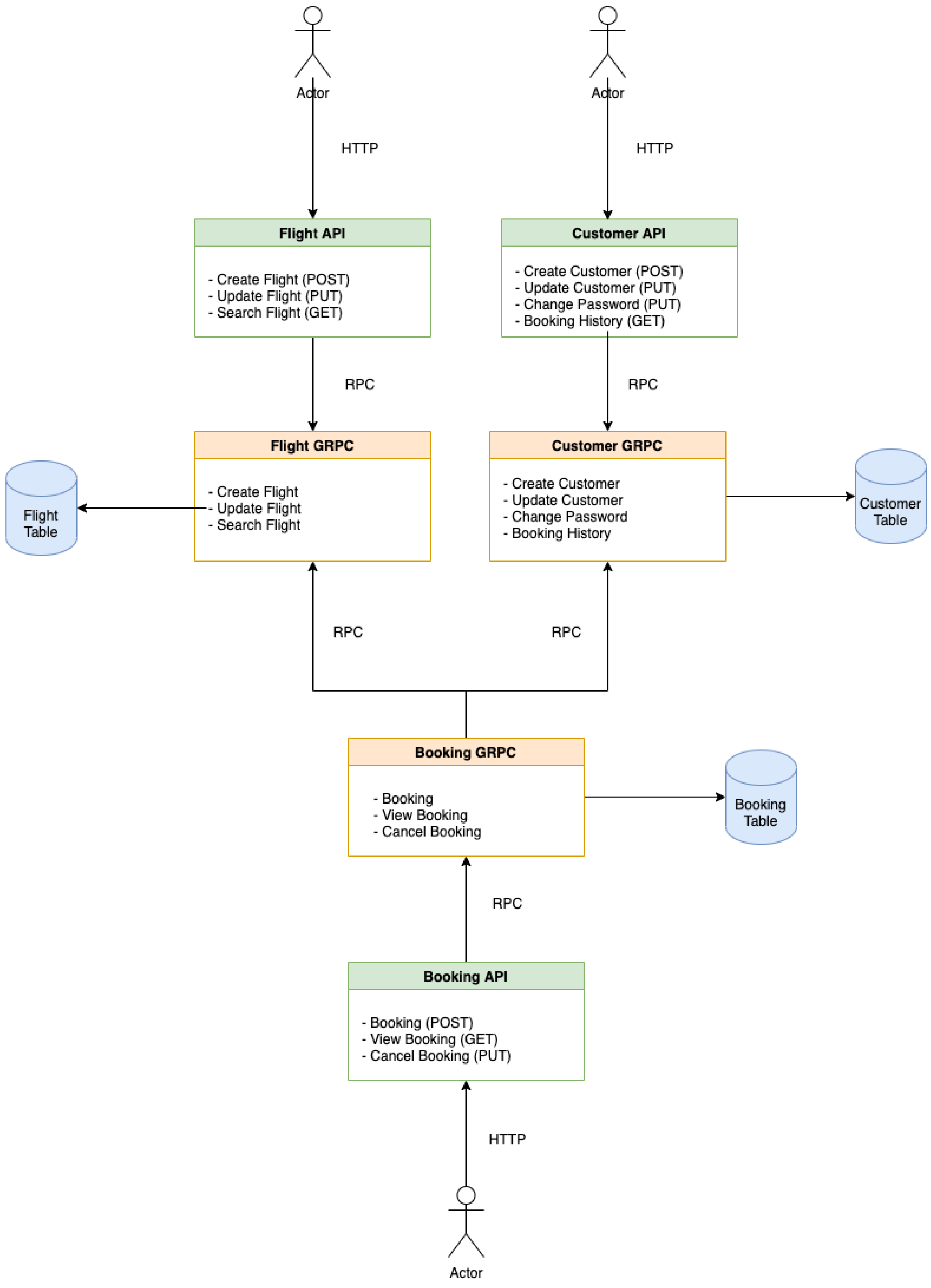
- Uses GRPC for your backend
- Uses REST API for your Public API
- Uses CockroachDB for data storage

Hint: You could learn from examples project (sample) to understand how to create your GRPC Server and REST API.

The booking microservices includes three main services consists:

- Booking Service: uses for booking includes:
  - Booking
  - View Booking
  - Cancel Booking
- Flight Service: uses for manage flight
  - Create Flight
  - Update Flight
  - Search Flight
- Customer Service: uses for manage customer
  - Create Customer
  - Update Customer
  - Change Password
  - Booking History

High Level Design



- All APIs will expose to client via REST (HTTP)
- All internal services will communicate via RPC

- There are 3 REST API servers and 3 GRPC servers
- What will API server do? -> API will only take the request from the client, and validate the request, then communicate to GRPC Server to handle business, after received the result from GRPC Server, the API server need transform the data and return to client.
- What will GRPC server do? -> GRPC server will provide all business services need, it communicate to database to handle query.

Basic Design

- Booking Service:
  - Booking: To create a new booking, you need the customer information (customer must be exist in the database) and flight information too. The flight must be validated, not late before 12 hours, assume the max flight slots only have 100, therefore your also need to validate the slot too. The customer will not allow to booking if all slots are full filled. When create a new booking successful, the booking service should return a booking code for customer to check later.

  Sample API response could be:

  ```
  {
      "code": 201,
      "payload": {
          "booking_code": "JWS3AZ",
          "booking_date": 21/06/2021,
          "customer_id": "2314295819825",
          "flight_id": "2132451255124",
          "status": "Success"
      }
  }
  ```

  - View Booking: After the customer have the booking code, he/she could uses it to find the booking information. The booking service will ask customer service to find the customer information, as well as flight service to find the flight information base on the ID. Assume when the booking code is found in database, it will need more information of customer and flight to return to the client.

  Sample API response could be:

  ```
  {
      "code": 200,
      "payload": {
          "booking_code": "JWS3AZ",
          "booking_date": 21/06/2021,
          "customer": {
              "id": "2314295819825",
              "name": "John Doe",
              "date_of_bith": "1/1/1970",
  ```

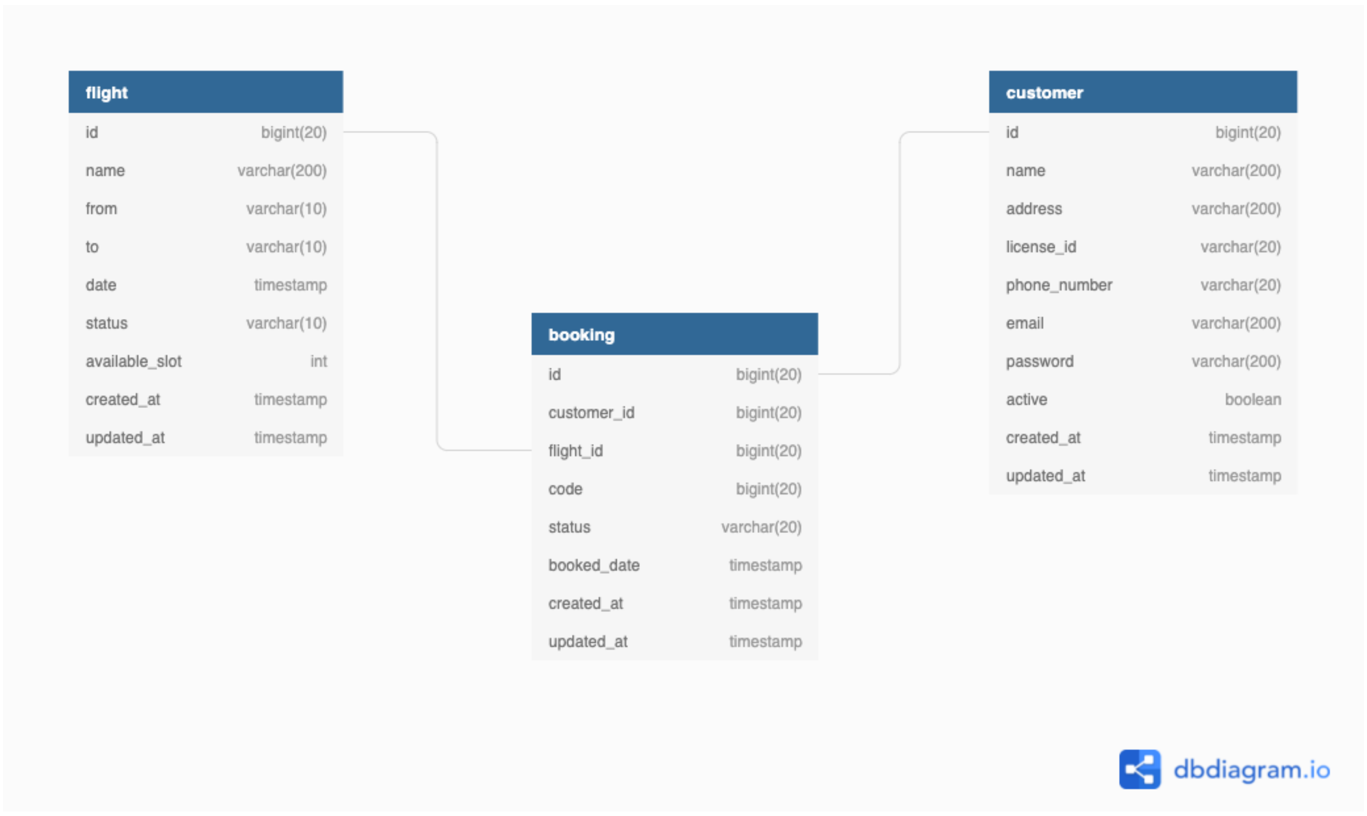
~~~
                "address": "FPT Software",
                "email": "john.doe@fsoft.com.vn"
            },
            "flight": {
                "id": "2132451255124",
                "from": "HAN",
                "to": "SGN",
                "depart_date": "25/06/2021",
                "depart_time": "07:00:00",
                "status": "Scheduled",
                "slot": "25A",
                "flight_plane": "VN212",
             }
        }
    }

- Cancel booking: The customer could cancel a booking not late than 48
hours before the flight take off. The service need to verify the
booking is valid before cancel it.


Sample API response could be:
```json
{
    "code": 204,
    "payload": {
        "booking_code": "JWS3AZ",
        "cancel_date": "23/06/2021",
        "customer_id": "2314295819825",
        "flight_id": "2132451255124",
        "status": "Cancel",
    }
}
~~~

- Flight Service:
    - Create Flight: Create the flight information includes the from place, and to destination, time,...
    - Update Flight: Update any information, such as status of flight, number of slot available,...
    - Search Flight: Search fight by from place and to destination, date,...
- Customer Service:
    - Create Customer: Create customer information includes name, address, date of birth, email,...
    - Update Customer: Update any information
    - Change Password: Change the old password to new password, please hash the password, not allowed to store it raw
    - Booking History: Search Booking History, show with pagging, sort by date.

Entity Relationship Diagram



flight	
id	bigint(20)
name	varchar(200)
from	varchar(10)
to	varchar(10)
date	timestamp
status	varchar(10)
available_slot	int
created_at	timestamp
updated_at	timestamp

booking	
id	bigint(20)
customer_id	bigint(20)
flight_id	bigint(20)
code	bigint(20)
status	varchar(20)
booked_date	timestamp
created_at	timestamp
updated_at	timestamp

customer	
id	bigint(20)
name	varchar(200)
address	varchar(200)
license_id	varchar(20)
phone_number	varchar(20)
email	varchar(200)
password	varchar(200)
active	boolean
created_at	timestamp
updated_at	timestamp

Ref:

- 1 Customer has many booking
- 1 Flight has many booking