ticket-booking-service

Implementation of a simple ticket service that facilitates the discovery, temporary hold, and final reservation of seats within a high-demand performance venue.

This application is developed using Spring Boot, Spring JDBC, Spring RESTful web services, Maven, HSQLDB.

Assumptions

---

1. Users are provided seats based on the availability.
2. No seat numbers.
3. Hold time for the seats is 60 seconds. If the user doesn't reserve the seats before 60 seconds, then the holds are removed and user has to send a request again to hold the seats.
4. No notification for the expiration of seat holds.
5. User can hold and reserve the seats at multiple levels by providing the minLevel and maxLevel.

Building Project

---

1. Clone the project

```
2. git clone https://github.com/vamshins/ticket-booking-service.git
```

3. Kindly make sure JAVA_HOME environment variable is configured and maven bin directory is added to PATH environment variable. Run the following commands

```
4. cd ticket-booking-service
5. mvn package
6. cd target
7. java -jar ticket-booking-service-0.0.1-SNAPSHOT.jar
```

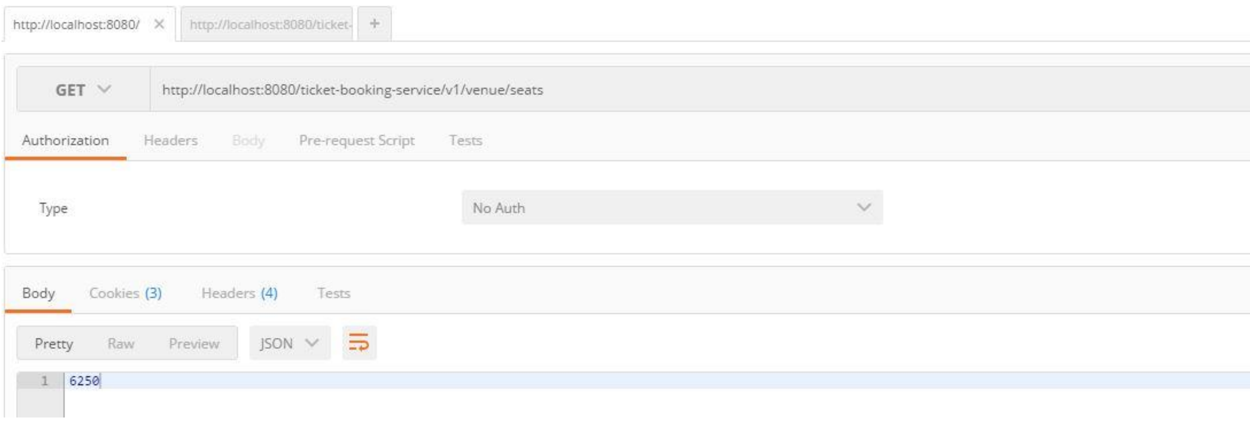
After running the above commands successfully, you should see the following messages.

```
.... Tomcat started on port(s): 8080 (http)
.... Started TicketBookingServiceApplication in 8.837 seconds (JVM running for
9.214)
```

RESTful Web Services

1. Find the number of seats available within the venue, optionally by seating level
   Note: available seats are seats that are neither held nor reserved.

   - Total seats available in all venues:
   - `GET - http://localhost:8080/ticket-booking-service/v1/venue/seats`

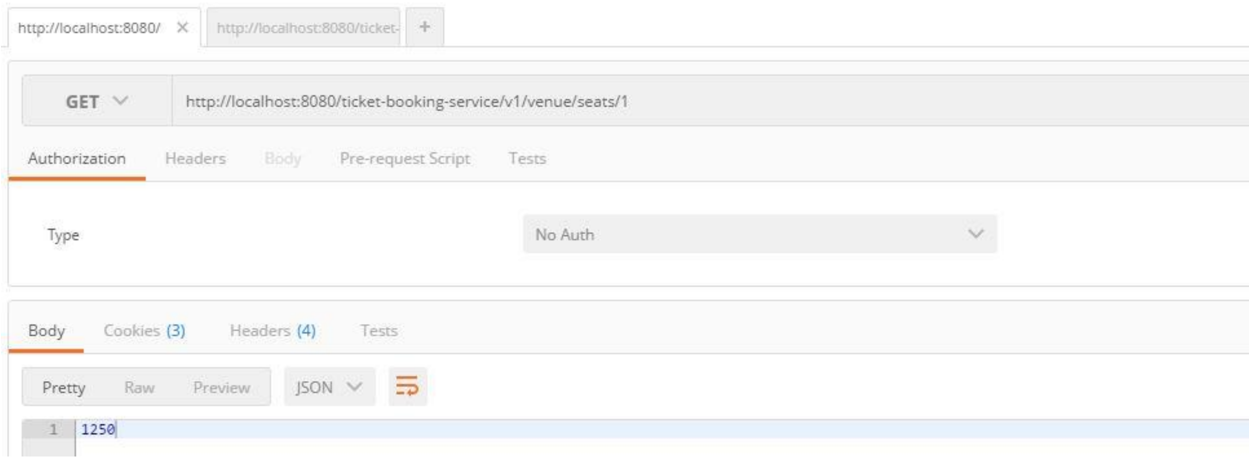


http://localhost:8080/ticket-booking-service/v1/venue/seats

Type: No Auth

```
6250
```

   - Number of seats available at a particular level:
   - `GET - http://localhost:8080/ticket-booking-service/v1/venue/seats/{levelId}`



```
http://localhost:8080/ticket-booking-service/v1/venue/seats/1
```

Type: No Auth

```
1250
```

2. Find and hold the best available seats on behalf of a customer, potentially limited to specific levels Note: each ticket hold should expire within a set number of seconds.

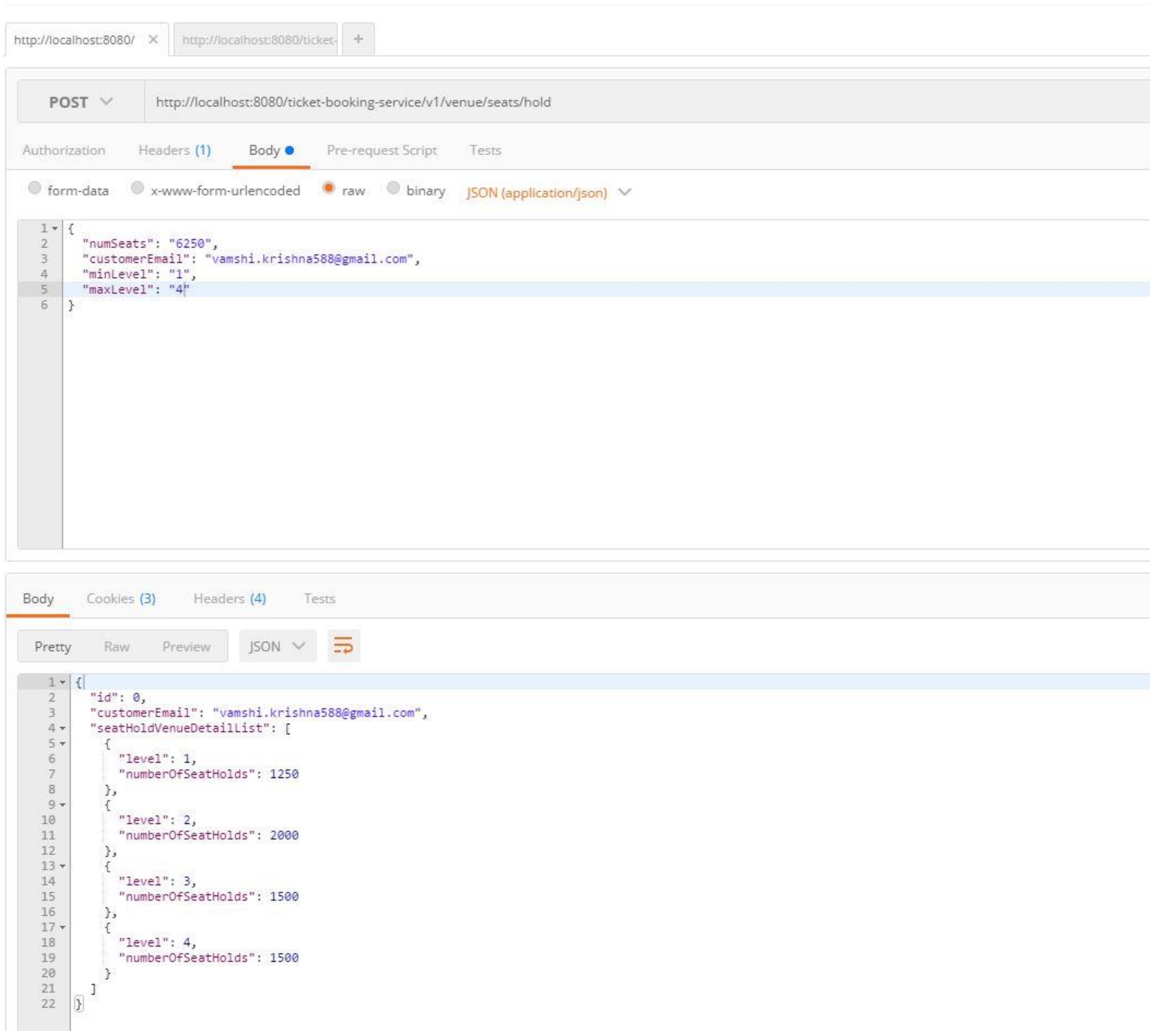
```
3. POST - http://localhost:8080/ticket-booking-service/v1/venue/seats/hold
```

RequestBody:

```
{
  "numSeats": "6250",
  "customerEmail": "vamshi.krishna588@gmail.com",
  "minLevel": "1",
  "maxLevel": "4"
}
```

ResponseEntity:

```
{
  "id": 1,
  "customerEmail": "vamshi.krishna588@gmail.com",
  "seatHoldVenueDetailList": [
    {
      "level": 1,
      "numberOfSeatHolds": 1250
    },
    {
      "level": 2,
      "numberOfSeatHolds": 2000
    },
    {
      "level": 3,
      "numberOfSeatHolds": 1500
    },
    {
      "level": 4,
      "numberOfSeatHolds": 1500
    }
  ]
}
```



http://localhost:8080/ticket-booking-service/v1/venue/seats/hold

```
{
  "numSeats": "6250",
  "customerEmail": "vamshi.krishna588@gmail.com",
  "minLevel": "1",
  "maxLevel": "4"
}
```

```
{
  "id": 0,
  "customerEmail": "vamshi.krishna588@gmail.com",
  "seatHoldVenueDetailList": [
    {
      "level": 1,
      "numberOfSeatHolds": 1250
    },
    {
      "level": 2,
      "numberOfSeatHolds": 2000
    },
    {
      "level": 3,
      "numberOfSeatHolds": 1500
    },
    {
      "level": 4,
      "numberOfSeatHolds": 1500
    }
  ]
}
```

This request will expire after 60 seconds. Before that, user has to reserve the seats using the web service in the following request.

4. Reserve and commit a specific group of held seats for a customer

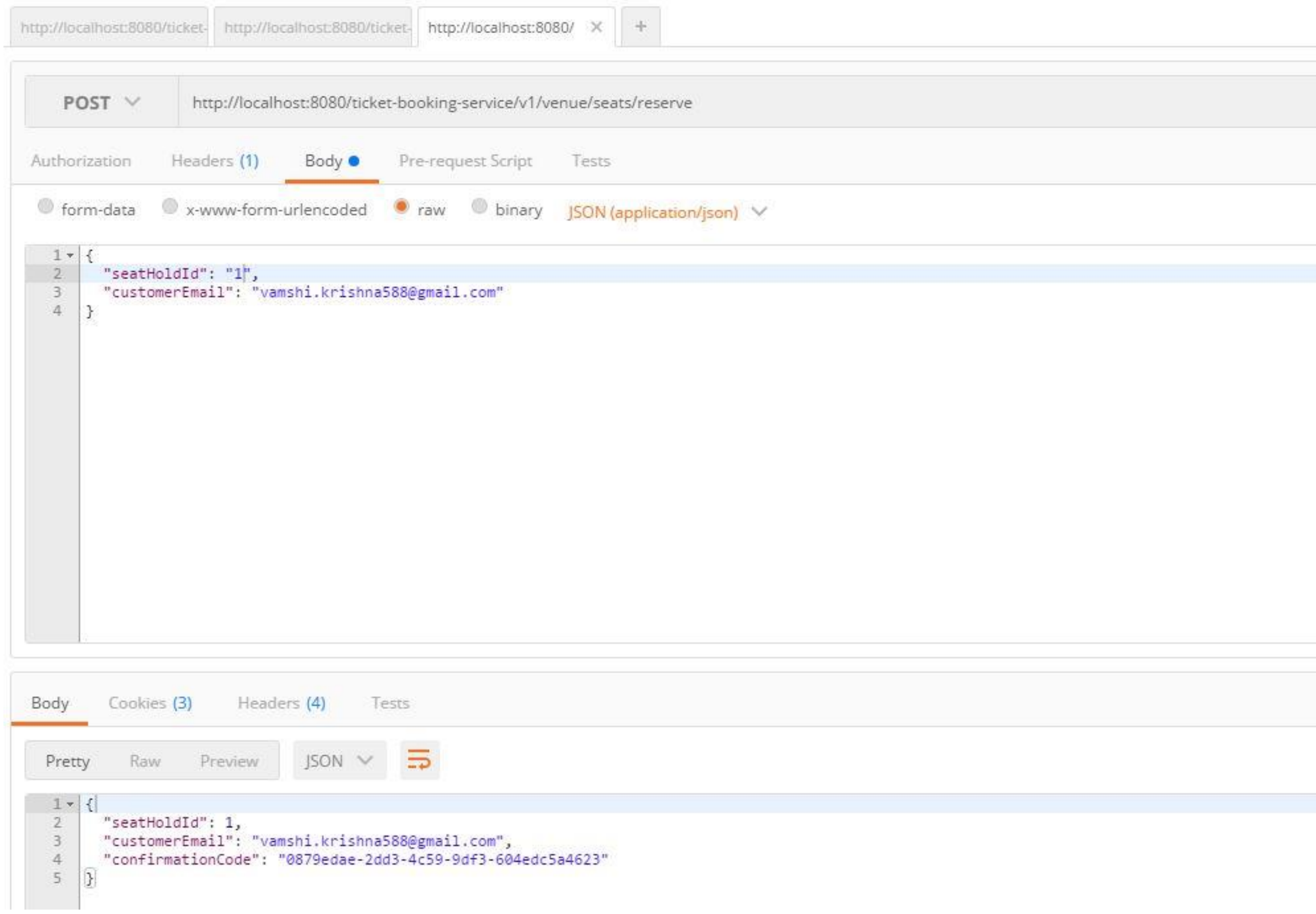
```
5. POST - http://localhost:8080/ticket-booking-service/v1/venue/seats/reserve
```

RequestBody:

```
{
  "seatHoldId": "1",
  "customerEmail": "vamshi.krishna588@gmail.com"
}
```

ResponseEntity:

```
{
  "seatHoldId": 1,
  "customerEmail": "vamshi.krishna588@gmail.com",
  "confirmationCode": "0879edae-2dd3-4c59-9df3-604edc5a4623"
}
```



POST http://localhost:8080/ticket-booking-service/v1/venue/seats/reserve

raw JSON (application/json)

```
{
  "seatHoldId": "1",
  "customerEmail": "vamshi.krishna588@gmail.com"
}
```

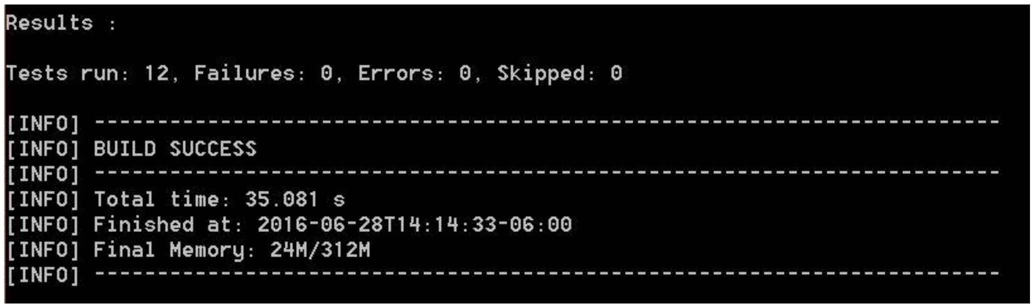
JSON

```
{
  "seatHoldId": 1,
  "customerEmail": "vamshi.krishna588@gmail.com",
  "confirmationCode": "0879edae-2dd3-4c59-9df3-604edc5a4623"
}
```

Testing Results

Tests are done using JUnit. Tests are run using the command

```
mvn test
```



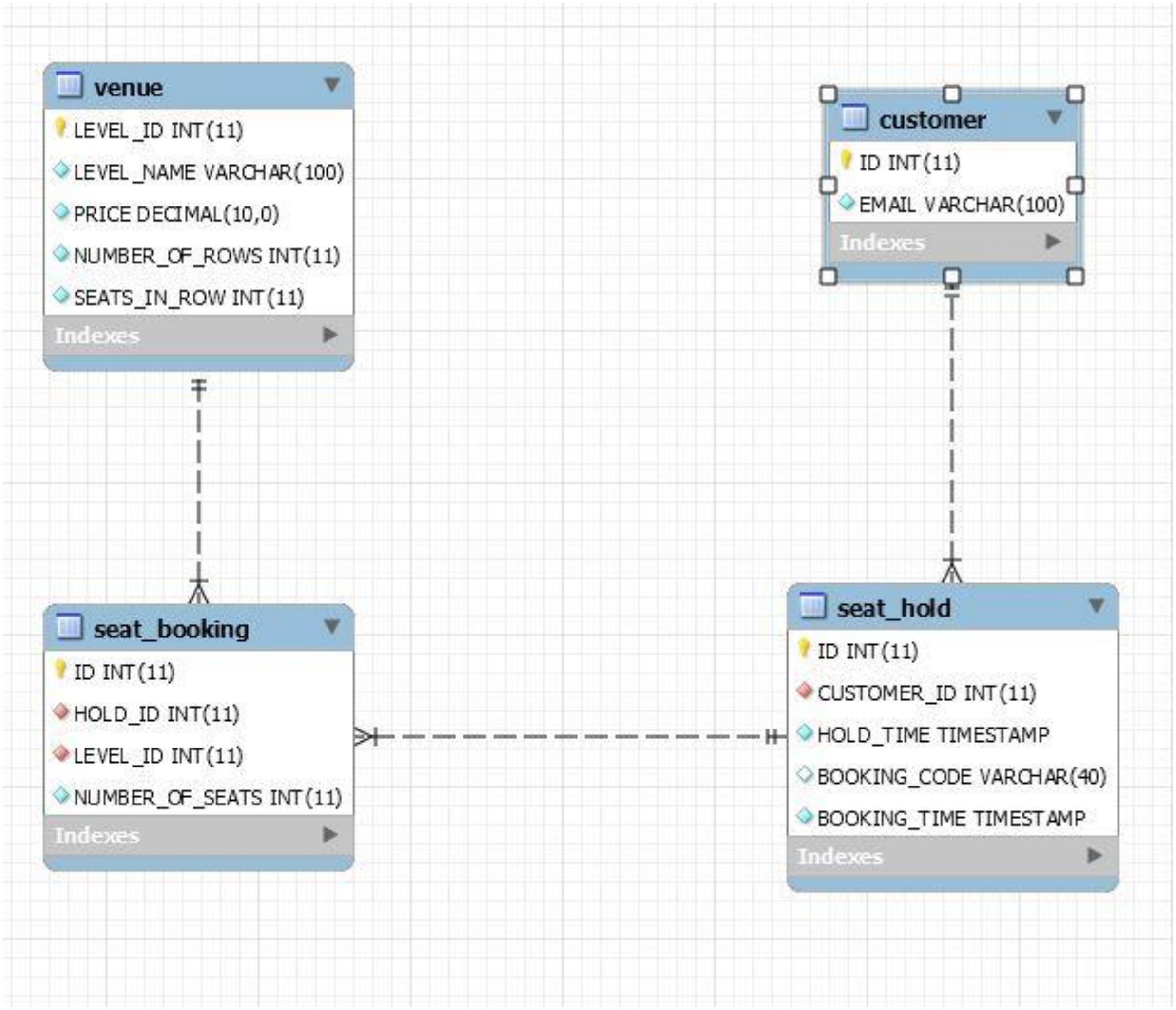
```
Results :

Tests run: 12, Failures: 0, Errors: 0, Skipped: 0

[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 35.081 s
[INFO] Finished at: 2016-06-28T14:14:33-06:00
[INFO] Final Memory: 24M/312M
[INFO] ------------------------------------------------------------------------
```

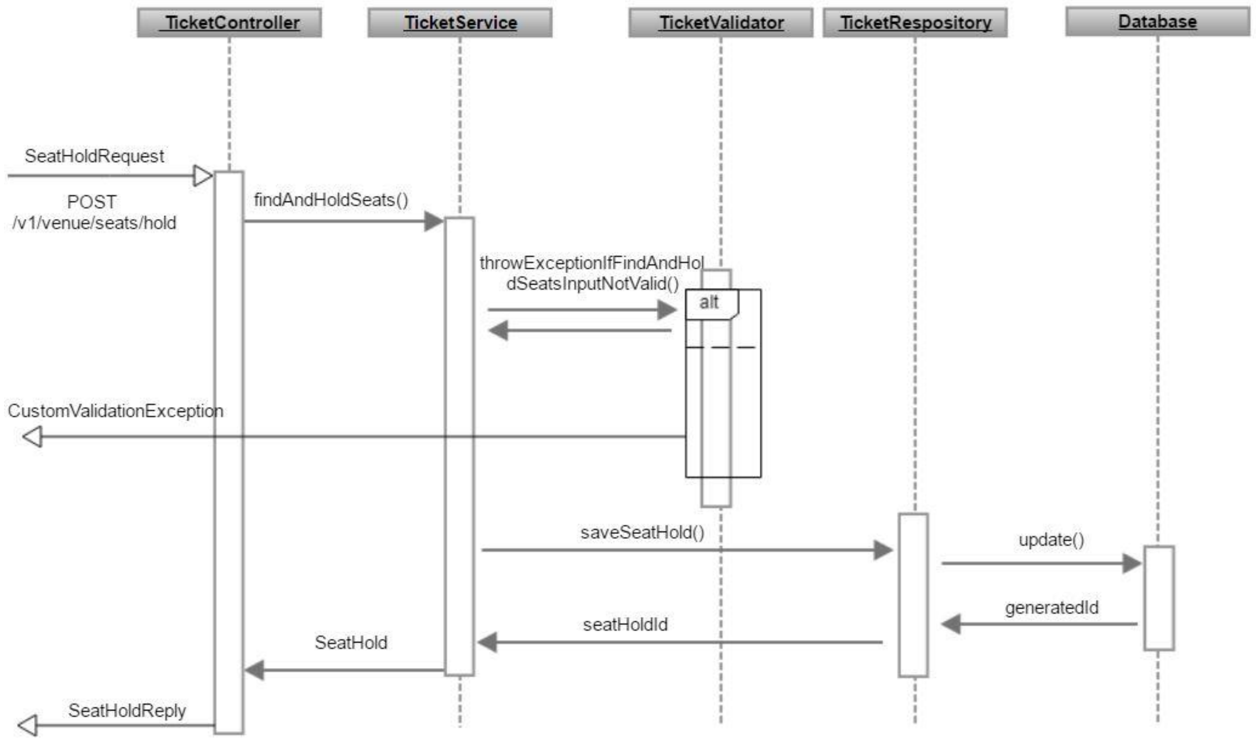
DB Schema

The application is designed using HSQLDB. For illustration purposes, I have generated the schema design using MySQL Workbench.



Sequence Diagram for holding seats

The following diagram shows the request and response flow of holding the seats.


TicketController
TicketService
TicketValidator
TicketRespository
Database
SeatHoldRequest
POST /v1/venue/seats/hold
findAndHoldSeats()
throwExceptionIfFindAndHoldSeatsInputNotValid()
alt
CustomValidationException
saveSeatHold()
update()
generatedId
seatHoldId
SeatHold
SeatHoldReply