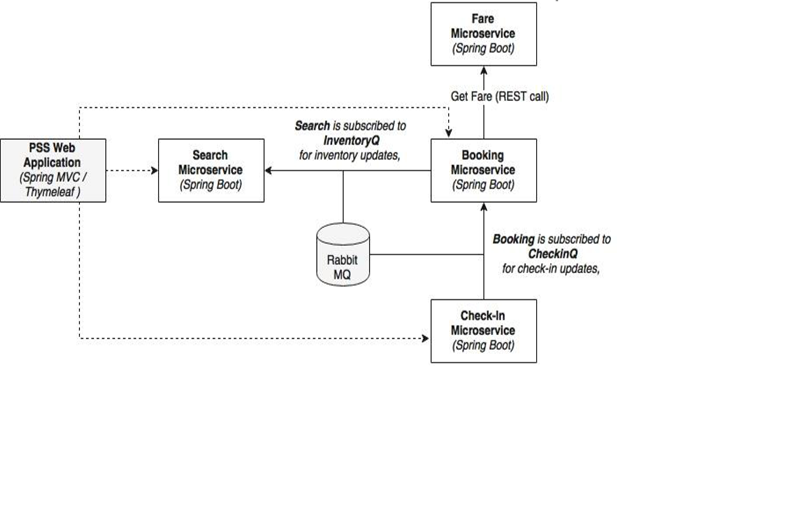
**MICROSERVICES EVOLUTION – A CASE STUDY**

We will understand the case study of BrownField Airline and their journey from a monolithic **Passenger Sales and Service** (PSS) application to a next generation microservices architecture by adhering to the principles and practices that were discussed before.

**Reviewing the microservices capability model**

We are about to implement four microservices such as Fare, Search, Booking and Check-in. In order to test the application, there is a website application developed using Spring MVC with Thymeleaf templates (needed for html pages). The asynchronous messaging is implemented with the help of RabbitMQ. In this implementation, the **Oracle database will be used with separate schema for each Microservice**.



**The following steps are used to setup PSS Microservices project**:

1. Create tablespaces, schemas, tables, sequences and insert data by referring ‘Documents/Misc/

Airline\_PSS\_Schema.doc’ file.

1. Download STS from [https://spring.io/tools/sts/all.](https://spring.io/tools/sts/all) Start STS (Spring Tool Suite) and select

‘**MicroservicesWorkspace**’ from the backup.

1. Start FaresFlightTickets by right click and Run as **Spring Boot App**.
2. Install **RabbitMQ** Server from Softwares folder. After installation check service status in start-> run -> services.msc

**Observation**: Status: Running, Startup type: Automatic

**Note**: The pre-requisite for RabbitMQ is **Erlang**. Hence Install OTP\_win64\_19.3.exe from Softwaresfolder.

1. Start SearchFlightTickets by right click and Run as **Spring Boot App**.
2. Start BookingFlightTickets by right click and Run as Spring Boot App.
3. Start CheckInCustomers by right click and Run as Spring Boot App.
4. Start FlightWebSite by right click and Run as Spring Boot App.

Each service has multiple packages and their purposes are explained as follows:

1. The entity package contains the JPA entity classes for mapping to the database tables.
2. The repository package contains repository classes, which are based on Spring Data JPA.
3. The component package hosts all the service components where the business logic is implemented.
4. The controller package hosts the **REST endpoints** and the **Messaging endpoints**. Controller classes internally utilize the component classes for execution.
5. The root package (com.brownfield.pss.fares) contains the default Spring Boot application.

The below table contains service endpoints and communication styles:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Microservice Name** |  | **REST endpoints** | | | | | |  |  | **Messaging Endpoints** |  | **Used By** |
|  |  | **(synchronous)** | | | | | |  |  | **(asynchronous)** |  |  |
|  |  |  |  |  |  |
| FareFlightTickets |  | <http://localhost:8081/fares/get> |  | | | | |  |  |  |  | Booking microservice |
|  |  |  | | |  |  |  |  |  |  |  |  |
| SearchFlightTickets |  | <http://localhost:8090/search/get> | | | | | |  |  |  |  | Website |
|  |  |  |  |  |  |  |  |  |  | |  |  |
| SearchFlightTickets |  |  |  |  |  |  |  |  | @RabbitListener(queues = "**SearchQ**") | |  | Search microservice itself |
|  |  |  |  |  |  |  |  |  |  |  |  | subscribed to **SearchQ** |
|  |  |  |  |  |  |  |  |  |  |  |  | for inventory updates. |
|  |  |  | | |  | |  |  |  |  |  |  |
| BookingFlightTickets |  | <http://localhost:8060/booking/create> | | | |  | |  |  |  |  | Website |
|  |  |  | | |  | | |  |  |  |  |  |
| BookingFlightTickets |  | [http://localhost:8060/booking/get/{id}](http://localhost:8060/booking/get/%7bid%7d) | | | | | |  |  |  |  | Checkin, website |
|  |  |  |  |  |  |  |  |  |  | |  |  |
| BookingFlightTickets |  |  |  |  |  |  |  |  | template.convertAndSend("**SearchQ**", | |  | Search Microservice |
|  |  |  |  |  |  |  |  |  | message); | |  |  |
|  |  |  |  |  |  |  |  |  |  | |  |  |
| BookingFlightTickets |  |  |  |  |  |  |  |  | @RabbitListener(queues = | |  | Booking service |
|  |  |  |  |  |  |  |  |  | "**CheckINQ**") | |  | subscribed to **CheckINQ** |
|  |  |  |  |  |  |  |  |  |  |  |  | for check-in updates. |
|  |  |  | | | | | |  |  |  |  |  |
| CheckInCustomers |  | <http://localhost:8070/checkin/create> | | |  | | |  |  |  |  | Website |
|  |  |  | | | | | |  |  |  |  |  |
| CheckInCustomers |  | http://localhost:8070/checkin/get/{id} | | | | | |  |  |  |  | Not used |
|  |  |  |  |  |  |  |  |  |  | |  |  |
| CheckInCustomers |  |  |  |  |  |  |  |  | template.convertAndSend("**CheckINQ** | |  | Booking Microservice |
|  |  |  |  |  |  |  |  |  | ", message); | |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |

We have accomplished the following items in our microservice implementation so far:

1. Each microservice exposes a set of REST/JSON endpoints for accessing business capabilities
2. Each microservice implements certain business functions using the Spring framework.
3. Each microservice has its own schema in Oracle database.
4. Microservices are built with Spring Boot, which has an embedded Tomcat server as the HTTP listener.
5. RabbitMQ is used as an external messaging service. Search, Booking, and Check-in interact with each other through asynchronous messaging.
6. An OAuth2-based security mechanism is developed to protect the Microservices.

**DataBase Design**



**CREATING FARE SCHEMA**

Step 1: Connect to database

C:\>sqlplus system/manager@xe

Step2: Create tablespace

CREATE **TABLESPACE** tbs\_fareuser DATAFILE 'tbs\_fareuser.dat' SIZE 1M AUTOEXTEND ON;

Note: alter session set "\_ORACLE\_SCRIPT"=true; This is required in Oracle 12c

Step3: Create a new user in Oracle

CREATE **USER** fareuser IDENTIFIED BY aspire123 DEFAULT TABLESPACE tbs\_fareuser QUOTA unlimited on tbs\_fareuser;

**Note**: In oracle, a schema is created when a user is created.

Step4: Grant permissions

GRANT create session TO fareuser;

GRANT create table TO fareuser;

GRANT create sequence TO fareuser;

Step5: Disconnect from system account and connect to fareuser Sql>exit

C:\>sqlplus fareuser/aspire123@xe

Step6: Create tables and sequences

drop table fare cascade constraints; drop sequence fare\_seq;

create table fare (id number(19) primary key, fare varchar2(255), flight\_date varchar2(255), flight\_number varchar2(255));

create sequence fare\_seq start with 1 increment by 1;

Step7: Insert records

insert into fare(id, fare, flight\_date, flight\_number) values (fare\_seq.nextVal, '100', '22-JAN-16', 'BF100');

insert into fare(id, fare, flight\_date, flight\_number) values (fare\_seq.nextVal, '101', '22-JAN-16', 'BF101');

insert into fare(id, fare, flight\_date, flight\_number) values (fare\_seq.nextVal, '102', '22-JAN-16', 'BF102');



insert into fare(id, fare, flight\_date, flight\_number) values (fare\_seq.nextVal, '103', '22-JAN-16', 'BF103');

insert into fare(id, fare, flight\_date, flight\_number) values (fare\_seq.nextVal, '104', '22-JAN-16', 'BF104');

insert into fare(id, fare, flight\_date, flight\_number) values (fare\_seq.nextVal, '105', '22-JAN-16', 'BF105');

insert into fare values (fare\_seq.nextVal, '106', '22-JAN-16', 'BF106');

commit;

Step8: Read data from FAREUSER schema

SELECT \* FROM "FAREUSER"."FARE";

|  |  |  |  |
| --- | --- | --- | --- |
| ID | FLIGHT\_NUMBER | FLIGHT\_DATE | FARE |
| 1 | BF100 | 22-JAN-16 | 100 |
| 2 | BF101 | 22-JAN-16 | 101 |
| 3 | BF102 | 22-JAN-16 | 102 |
| 4 | BF103 | 22-JAN-16 | 103 |
| 5 | BF104 | 22-JAN-16 | 104 |
| 6 | BF105 | 22-JAN-16 | 105 |
| 7 | BF106 | 22-JAN-16 | 106 |

**CREATING SEARCH SCHEMA**

Step 1: Connect to database (ignore if already connected)

C:\>sqlplus system/manager@xe

Step2: Create tablespace

CREATE TABLESPACE tbs\_searchuser DATAFILE 'tbs\_searchuser.dat'

SIZE 1M AUTOEXTEND ON;

Note: alter session set "\_ORACLE\_SCRIPT"=true; This is required in Oracle 12c

Step3: Create a new user in Oracle

CREATE USER searchuser IDENTIFIED BY aspire123 DEFAULT TABLESPACE tbs\_searchuser QUOTA unlimited on tbs\_searchuser;

Note: In oracle, a schema is automatically created when a user is created.

Step4: Grant permissions

GRANT create session TO searchuser;

GRANT create table TO searchuser;

GRANT create sequence TO searchuser;

2



Step5: Disconnect from system account and connect to searchuser Sql>exit

C:\>sqlplus searchuser/aspire123@xe

Step6: Create tables and sequences

drop table fare cascade constraints; drop table inventory cascade constraints; drop table flight cascade constraints;

drop sequence fare\_seq;

drop sequence flight\_seq;

drop sequence inventory\_seq;

create sequence fare\_seq start with 1 increment by 1; create sequence flight\_seq start with 1 increment by 1; create sequence inventory\_seq start with 1 increment by 1;

create table **fare** (fare\_id number(19) primary key, currency varchar2(255), fare varchar2(255));

create table **inventory** (inv\_id number(19) primary key, count number(10) not null);

create table **flight** (id number(19) primary key, origin varchar2(255), destination varchar2(255), flight\_number varchar2(255), flight\_date varchar2(255),

fare\_id number(19) references fare(fare\_id), inv\_id number(19) references inventory(inv\_id));

Step7: Insert records

insert into fare (currency, fare, fare\_id) values ('USD', 100, fare\_seq.nextVal);

insert into fares (currency, fare, fare\_id) values ('USD', 101, fare\_seq.nextVal);

insert into fare (currency, fare, fare\_id) values ('USD', 102, fare\_seq.nextVal);

insert into fare (currency, fare, fare\_id) values ('USD', 103, fare\_seq.nextVal);

insert into fare (currency, fare, fare\_id) values ('USD', 104, fare\_seq.nextVal);

insert into fare (currency, fare, fare\_id) values ('USD', 105, fare\_seq.nextVal);

3



insert into fare (currency, fare, fare\_id) values ('USD', 106, fare\_seq.nextVal);

insert into inventory (count, inv\_id) values (100, inventory\_seq.nextVal);

insert into inventory (count, inv\_id) values (100, inventory\_seq.nextVal);

insert into inventory (count, inv\_id) values (100, inventory\_seq.nextVal);

insert into inventory (count, inv\_id) values (100, inventory\_seq.nextVal);

insert into inventory (count, inv\_id) values (100, inventory\_seq.nextVal);

insert into inventory (count, inv\_id) values (100, inventory\_seq.nextVal);

insert into inventory (count, inv\_id) values (100, inventory\_seq.nextVal);

insert into flight (id, flight\_number, origin, destination, flight\_date, fare\_id, inv\_id) values (flight\_seq.nextVal, 'BF100', 'SEA', 'SFO', '22-JAN-16', 1, 1);

insert into flight (id, flight\_number, origin, destination, flight\_date, fare\_id, inv\_id) values (flight\_seq.nextVal, 'BF101', 'NYC', 'SFO', '22-JAN-16', 2, 2);

insert into flight (id, flight\_number, origin, destination, flight\_date, fare\_id, inv\_id) values (flight\_seq.nextVal, 'BF102', 'CHI', 'SFO', '22-JAN-16', 3, 3);

insert into flight (id, flight\_number, origin, destination, flight\_date, fare\_id, inv\_id) values (flight\_seq.nextVal, 'BF103', 'HOU', 'SFO', '22-JAN-16', 4, 4);

insert into flight (id, flight\_number, origin, destination, flight\_date, fare\_id, inv\_id) values (flight\_seq.nextVal, 'BF104', 'LAX', 'SFO', '22-JAN-16', 5, 5);

insert into flight (id, flight\_number, origin, destination, flight\_date, fare\_id, inv\_id) values (flight\_seq.nextVal, 'BF105', 'NYC', 'SFO', '22-JAN-16', 6, 6);

insert into flight (id, flight\_number, origin, destination, flight\_date, fare\_id, inv\_id) values (flight\_seq.nextVal, 'BF106', 'NYC', 'SFO', '22-JAN-16', 7, 7);

commit;

Step8: Read data from SEARCHUSER schema SELECT \* FROM "SEARCHUSER"."FARE";

4



|  |  |  |
| --- | --- | --- |
| FARE\_ID | FARE | CURRENCY |
| 1 | 100 | USD |
| 2 | 101 | USD |
| 3 | 102 | USD |
| 4 | 103 | USD |
| 5 | 104 | USD |
| 6 | 105 | USD |
| 7 | 106 | USD |

SELECT \* FROM "SEARCHUSER"."INVENTORY";

|  |  |
| --- | --- |
| INV\_ID | COUNT |
| 1 | 100 |
| 2 | 100 |
| 3 | 100 |
| 4 | 100 |
| 5 | 100 |
| 6 | 100 |
| 7 | 100 |

SELECT \* FROM "SEARCHUSER"."FLIGHT";

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | FLIGHT\_NUMBER | FLIGHT\_DATE | ORIGIN | DESTINATION | FARE\_ID | INV\_ID |
|  |  |  |  |  |  |  |
| 1 | BF100 | 22-JAN-16 | SEA | SFO | 1 | 1 |
| 2 | BF101 | 22-JAN-16 | NYC | SFO | 2 | 2 |
| 3 | BF102 | 22-JAN-16 | CHI | SFO | 3 | 3 |
| 4 | BF103 | 22-JAN-16 | HOU | SFO | 4 | 4 |
| 5 | BF104 | 22-JAN-16 | LAX | SFO | 5 | 5 |
| 6 | BF105 | 22-JAN-16 | NYC | SFO | 6 | 6 |
| 7 | BF106 | 22-JAN-16 | NYC | SFO | 7 | 7 |

**CREATING BOOKING SCHEMA**

Step 1: Connect to database (ignore if already connected)

C:\>sqlplus system/manager@xe

Step2: Create tablespace

CREATE TABLESPACE tbs\_bookinguser DATAFILE 'tbs\_bookinguser.dat'

SIZE 1M AUTOEXTEND ON;

Note: alter session set "\_ORACLE\_SCRIPT"=true; This is required in Oracle 12c

Step3: Create a new user in Oracle

CREATE USER bookinguser IDENTIFIED BY aspire123 DEFAULT TABLESPACE tbs\_bookinguser QUOTA unlimited on tbs\_bookinguser;

5



Note: In oracle, a schema is created when a user is created.

Step4: Grant permissions

GRANT create session TO bookinguser;

GRANT create table TO bookinguser;

GRANT create sequence TO bookinguser;

Step5: Disconnect from system account and connect to bookinguser Sql>exit

C:\>sqlplus bookinguser/aspire123@xe

Step6: Create tables and sequences

drop table booking\_record cascade constraints; drop table inventory cascade constraints; drop table passenger cascade constraints;

drop sequence booking\_seq;

drop sequence inventory\_seq;

drop sequence passenger\_seq;

create sequence booking\_seq start with 1 increment by 1; create sequence inventory\_seq start with 1 increment by 1; create sequence passenger\_seq start with 1 increment by 1;

create table booking\_record (id number(19) primary key, booking\_date timestamp, destination varchar2(255), fare varchar2(255), flight\_date varchar2(255), flight\_number varchar2(255), origin varchar2(255), **status** varchar2(255));

create table inventory (id number(19) primary key, available number(10) not null, flight\_date varchar2(255), flight\_number varchar2(255));

create table passenger (id number(19) primary key, first\_name varchar2(255), gender varchar2(255), last\_name varchar2(255), booking\_id number(19) references booking\_record(id));

Step7: **Insert records**

insert into inventory (flight\_number, flight\_date, available, id) values ('BF100', '22-JAN-16', 100, inventory\_seq.nextVal); insert into inventory (flight\_number, flight\_date, available, id) values ('BF101', '22-JAN-16', 100, inventory\_seq.nextVal); insert into inventory (flight\_number, flight\_date, available, id) values ('BF102', '22-JAN-16', 100, inventory\_seq.nextVal);

6



insert into inventory (flight\_number, flight\_date, available, id) values ('BF103', '22-JAN-16', 100, inventory\_seq.nextVal);

insert into inventory (flight\_number, flight\_date, available, id) values ('BF104', '22-JAN-16', 100, inventory\_seq.nextVal); insert into inventory (flight\_number, flight\_date, available, id) values ('BF105', '22-JAN-16', 100, inventory\_seq.nextVal); insert into inventory (flight\_number, flight\_date, available, id) values ('BF106', '22-JAN-16', 100, inventory\_seq.nextVal);

commit;

Step8: Read data from BOOKINGUSER schema

SELECT \* FROM "BOOKINGUSER"."INVENTORY";

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | FLIGHT\_NUMBER |  | FLIGHT\_DATE | AVAILABLE |
| 1 | BF100 |  | 22-JAN-16 | 100 |
| 2 | BF101 |  | 22-JAN-16 | **99** |
| 3 | BF102 |  | 22-JAN-16 | 100 |
| 4 | BF103 |  | 22-JAN-16 | 100 |
| 5 | BF104 |  | 22-JAN-16 | 100 |
| 6 | BF105 |  | 22-JAN-16 | 100 |
| 7 | BF106 |  | 22-JAN-16 | 100 |

SELECT \* FROM "BOOKINGUSER"."BOOKING\_RECORD";

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ID | BOOKING\_DATE | ORIGIN | DESTINATION | FARE | FLIGHT\_DATE | FLIGHT\_NUMBER | STATUS |
|  |  |  |  |  |  |  |  |
| 1 | 2017-06-06 | NYC | SFO | 101 | 22-JAN-16 | BF101 | **BOOKING\_CONFIRMED** |
|  | 20:46:01 |  |  |  |  |  |  |

SELECT \* FROM "BOOKINGUSER"."PASSENGER";

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | FIRST\_NAME | LAST\_NAME | GENDER | BOOKING\_ID |
| 1 | Gean | Franc | Male | 1 |

**CREATING CHECKIN SCHEMA**

Step 1: Connect to database (ignore if already connected)

C:\>sqlplus system/manager@xe

Step2: Create tablespace

CREATE TABLESPACE tbs\_checkinuser DATAFILE 'tbs\_checkinuser.dat'

SIZE 1M AUTOEXTEND ON;

Note: alter session set "\_ORACLE\_SCRIPT"=true; This is required in Oracle 12c

7



Step3: Create a new user in Oracle

CREATE USER checkinuser IDENTIFIED BY aspire123 DEFAULT TABLESPACE tbs\_checkinuser QUOTA unlimited on tbs\_checkinuser;

Note: In oracle a schema is created when a user is created.

Step4: Grant permissions

GRANT create session TO checkinuser;

GRANT create table TO checkinuser;

GRANT create sequence TO checkinuser;

Step5: Disconnect from system account and connect to checkinuser

Sql>exit

C:\>sqlplus checkinuser/aspire123@xe

Step6: Create tables and sequences

drop table check\_in\_record cascade constraints; drop sequence checkin\_seq;

create sequence checkin\_seq start with 1 increment by 1;

create table check\_in\_record (id number(19)primary key, booking\_id number(19) not null, check\_in\_time timestamp, first\_name varchar2(255), flight\_date varchar2(255), flight\_number varchar2(255), last\_name varchar2(255), **seat\_number** varchar2(255));

Step7: **Insert records**

No need to insert data manually

Step8: Read data from CHECKINUSER schema

SELECT \* FROM "CHECKINUSER"."CHECK\_IN\_RECORD";

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ID | BOOKING\_ID | CHECK\_IN\_TIME | FIRST\_ | LAST\_NAME | FLIGHT\_DATE | FLIGHT\_NUMBER | SEAT\_NUMBER |
|  |  |  | NAME |  |  |  |  |
|  |  |  |  |  |  |  |  |
| 1 | 1 | 2017-06-06 | Gean | Franc | 22-JAN-16 | BF101 | **28A** |
|  |  | 21:18:46 |  |  |  |  |  |

**Other useful commands**

DROP TABLESPACE tbs\_testuser INCLUDING CONTENTS AND DATAFILES;

DROP USER testuser;