Air Ticket Reservation System

Architecture & System Design

Author: Pedro Martín Gómez
Email: pmargom@gmail.com
Date: 12/06/2016

Introduction 3
Requirement analysis 3
  Architecture proposal 4
Data model presentation 5
  Entities 5
  Entity relations 5
Architecture diagrams 7
  Actors and their relations 7
  Use cases 7
  Sequence diagrams 9
    Public user use cases 9
    Airline user use cases 12
  Flow diagrams 14
UI and user interaction 14
Other comments 15

Introduction

The customer needs to create an Air Ticker Reservation System. The current document will describe the architecture and design of the system.

Requirement analysis

We have to model that allows users to make air ticket bookings. We have two kinds of user:

1. Public users.
2. Airline users.

Both of them have their own functionalities. In particular:

- A public user:
  - Are able to login using their Google+, Twitter LinkedIn accounts.
  - Are able to reserve air tickets searching from multiple flights based on multiple criteria.
  - Can make payments using credit cards.
  - Receive tickets as PDF via email. This email should contain useful links like ‘’My Bookings”, “Checkout Now”, “Cancel booking”, that will not require the user to login.
  - Are able to perform online checkout/cancellation, from 48 hrs and up to 4 hrs from departure, choose seats and print boarding passes.
  - Receive email alerts 48 hrs from departure, regarding their reservation, to encourage them for early online checkout. This email should contain a “Checkout Now” link that will not require the user to login.
- Airline staff:
  - Are able to login their directory credentials, even for public computers.
  - Receive reservation char as PDF at 1 hrs from departure.
  - Cancel any ticket, generating email alert and refund to public user.
  - Cancel the whole flight, generating email alerts and refunds to the public users.

Architecture proposal

After analyzing the customer requirements, my proposal like system architecture can be the following:

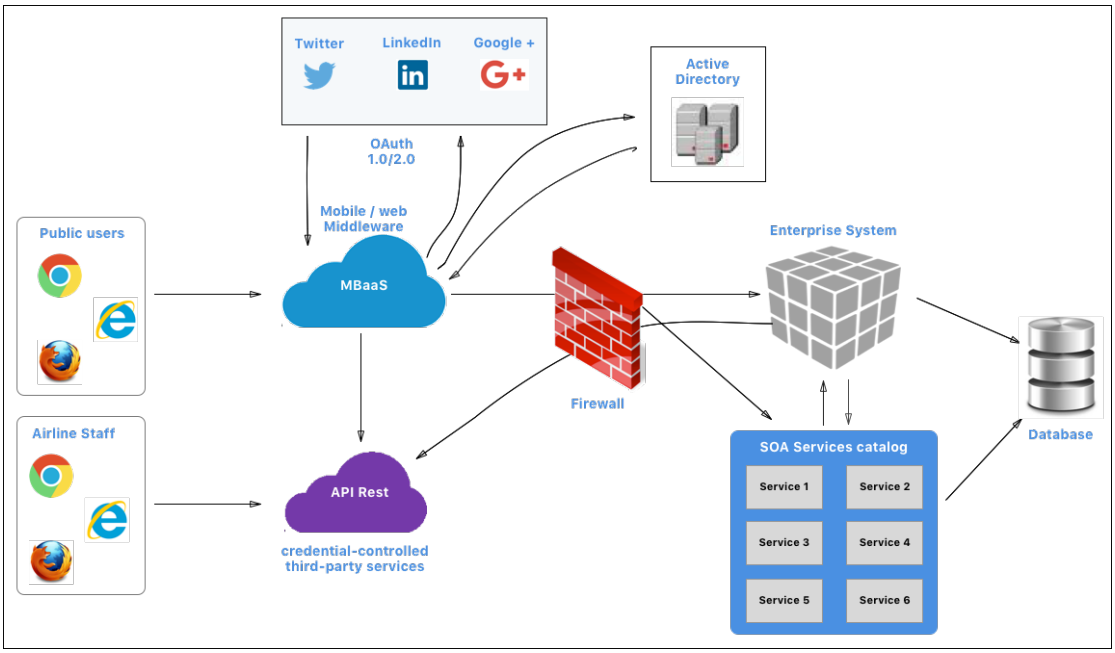


Figure 1: Air Ticket Reservation System Architecture

In the figure 1, it shows an architecture that combines different standard approaches. Specifically:

- SOA (service oriented architecture).
- MBaaS (mobile as a service architecture).
- API rest.

This “hybrid” approach will be compliant with the requirements indicated by the customer with low costs and high scalability in the future.

We have two fundamental components:

- MBaaS
- SOA

The first one will allow centralizing all login requests for both scenarios: active directory for staff users and social login for public users. This component not only will take care about login, in addition, it will offer other useful services like custom API’s, data storage, CDN, notifications and so on. Normally, this component will be present in cloud computing that is another requirement to accomplish.

The second one will allow the interoperability with the firewall-protected enterprise system by exposing a set of functionalities for authorized sessions (performed by the **middleware mobile/web component**).

In order to provide the communication between service catalog exposed (**SOA service catalog**) and the outside world, it was included an API that involves the set on endpoints for

user requests made by Internet browsers (**API credential-controlled third-party services component**).

Finally, thinking about the scalability:

- The architecture proposed will be ready for attending API requests from mobile apps (native and hybrid).
- New functionalities (social logins, data storages, CDN, etc. can be added extending the MBaaS platform).
- In the enterprise side, using SOA catalog services, new services can be added.

Data model presentation

This section describes the data model for the system. To simplify, they were made the following considerations:

- Users are able to reserve just direct flights, i.e., flights with no scales.
- Other entities for transaction logging history maintenance were omitted.

Entities

The list of entities is:

- **AirlineUsers**: information about airline users. In fact, only the necessary info to perform administration tasks as cancel a booking or the whole flight.
- **PublicUsers**: Information about public users who want to reserve flights.
- **PublicUserDetails**: Detail information about public users (phone, direction, cyty, etc.)
- **Airports**: Flight destinations. Needed to know the flight itinerary.
- **Airlines**: Airline companies who manage airplanes, reservations and emit boarding passes.
- **Aircrafts**: Aircrafts used by airlines. In this case, the system will be used by only one airline. However, the data model would support the management of several airlines.
- **Seats**: Set of seats, which will be occupied by passengers.
- **Flights**: Flights information managed by airlines.
- **Bookings**: Data that include flight, passengers, airport, airline, etc.
- **BordingPasses**: Similar information included in the reservation after checking in.
- **Payments**: Payment information that includes, flight, passengers, etc.
- **Passengers**: Information about people included in the booking by the public user.
- **Arrivals**: Information about airport destination, date and time of arrival.
- **Departures**: Information about airport source, date and time of departure.
- **CreditCards**: Information about credit cards used to pay bookings.
- **Notifications**: Messages to notify users about incidences (date/time modification, cancellation, etc.) related to the flight.

Entity relations

Basic relations identified:

- Reservations of flights.
- Seats involved in a flight.
- Date and time of a flight. Departures and arrivals.
- Airline desired.
- Itinerary of flights.
- Payments of flights.

- Passengers includes in a reservation..
- Public users and credit cards.
- Airline users admin operations.

So, we can rewrite above sentences like:

- A flight contains reservations, seats and has date and time.
- A flight belongs to an airline.
- A flight has itinerary.
- A public user has reservations and has one or more credit cards.
- A reservation can include more than one flight and more than a passenger.
- A payment is performed by a credit card.
- One or more payments can have been done with the same credit card.
- An airline user can modify or cancel any booking or the whole flight.
- A public user can pay

Below it shows the final data model diagram:

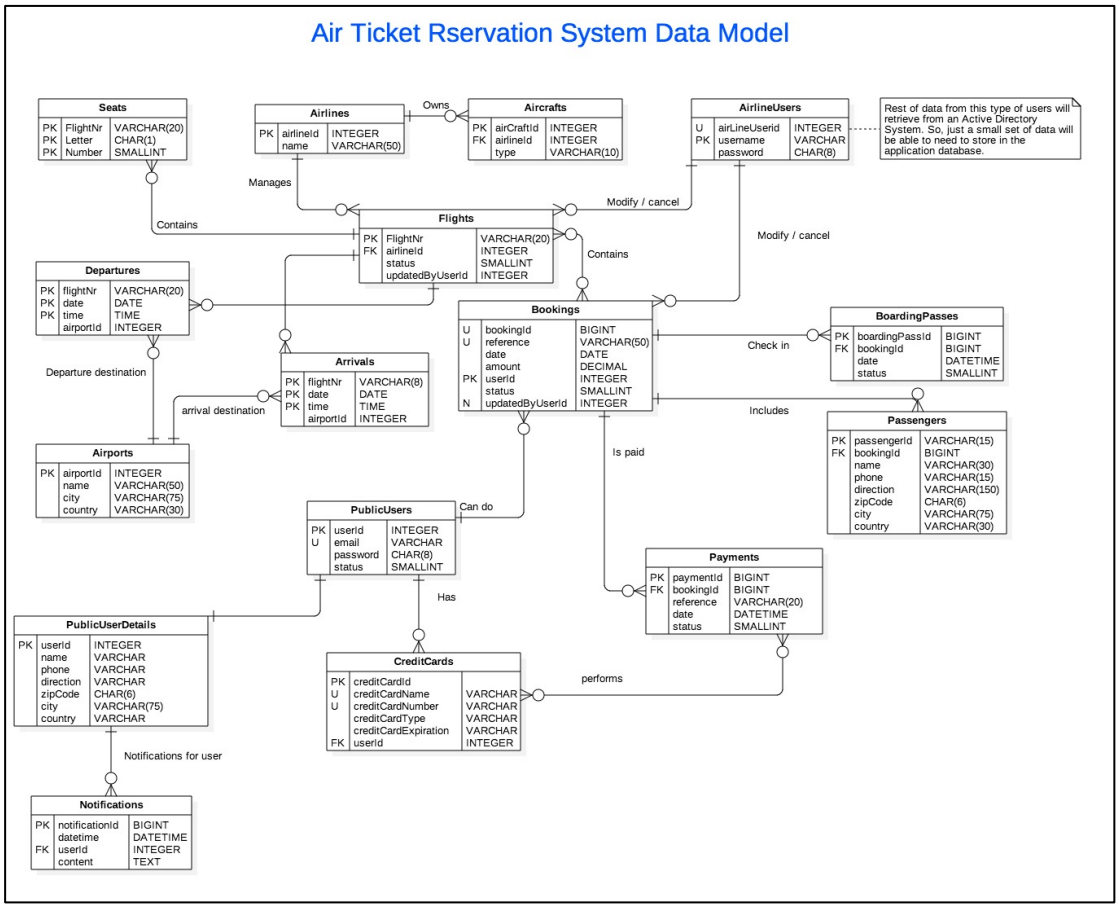


Figure 2: Data model diagram

Architecture diagrams

In this section, it shows different diagrams needed to model the booking system. These are:

- Actors and their relations.
- Use cases.
- Sequence.
- Flow.

Actors and their relations

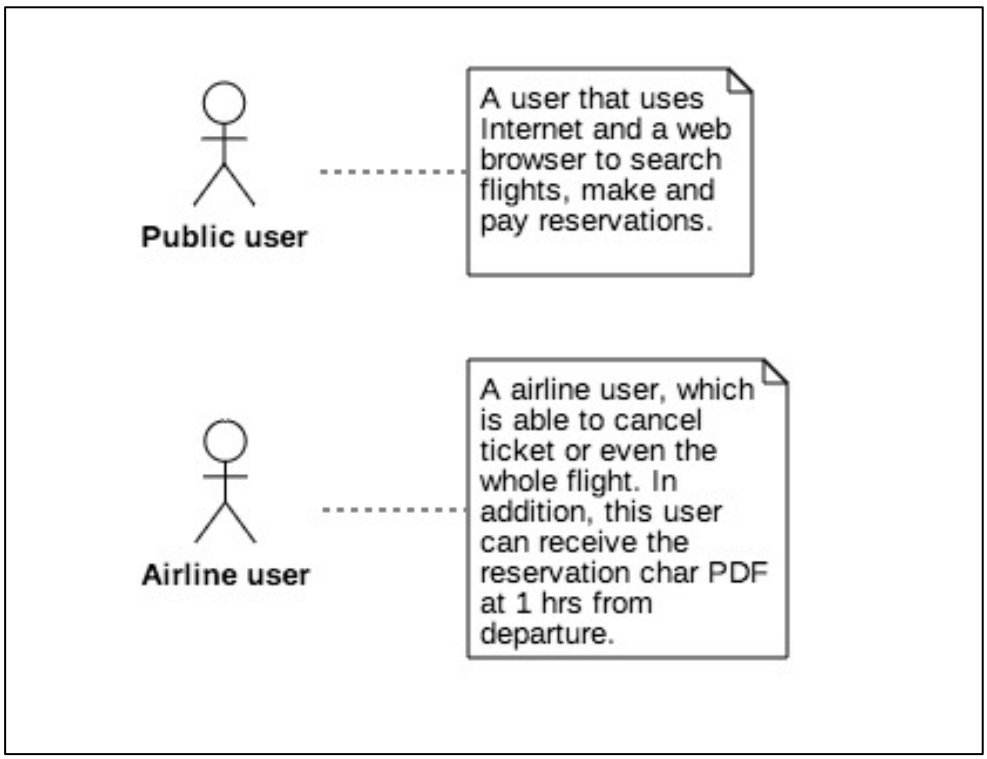


Figure 3: Actors and relations diagram

Use cases

As use cases, we have the following:

- A public user can:
  - Login.
  - Perform flights searching.
  - Managing their bookings:
    - Create a booking.
    - Cancel a booking.
    - Check out a booking.
    - Pay a booking:
      - Pay using a credit card.
- An airline user can:
  - Login.
  - Generate the reservation chart at 1hrs from departure.
  - Managing users bookings:
    - Cancel a ticket.
    - Cancel a whole flight.

Below, it shows the depicted diagram:

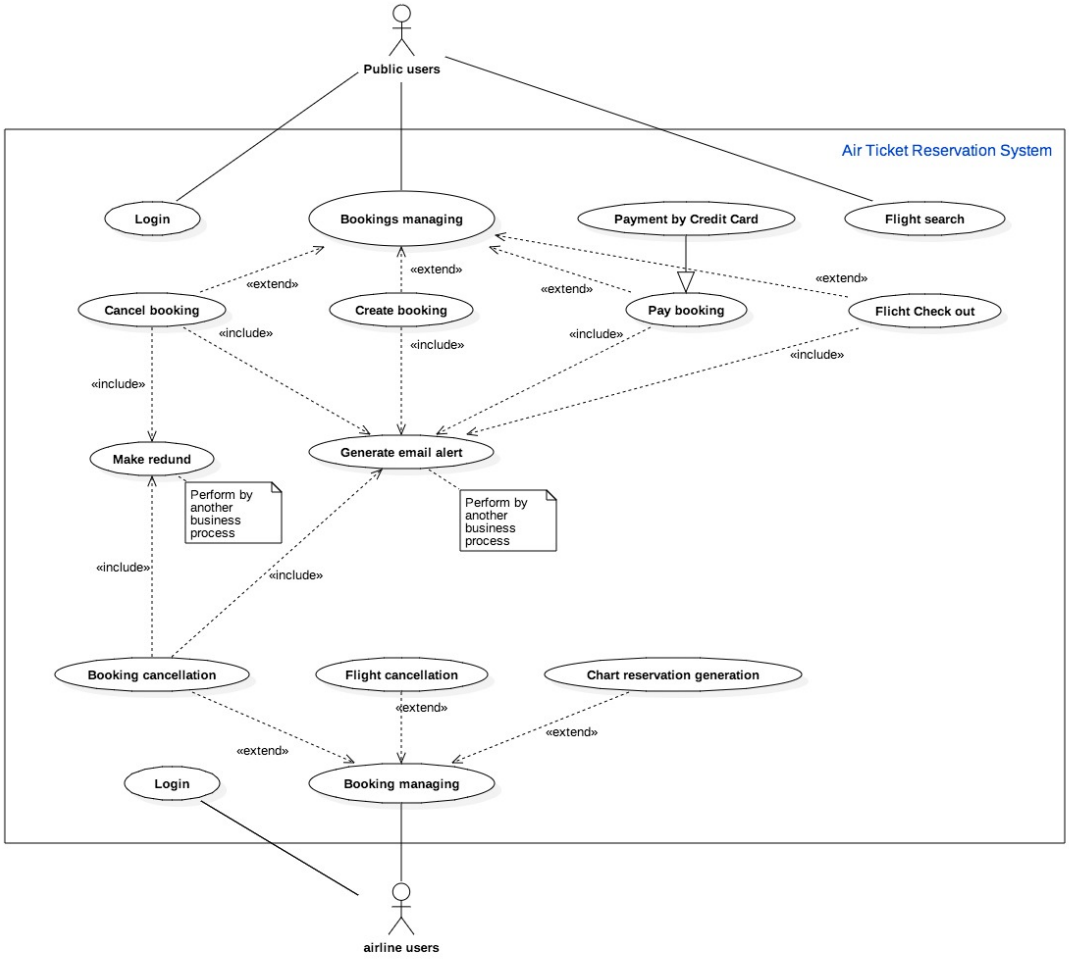


Figure 4: Use cases

Sequence diagrams

Following, it shows the sequence diagram for each use case.

Public user use cases

Login

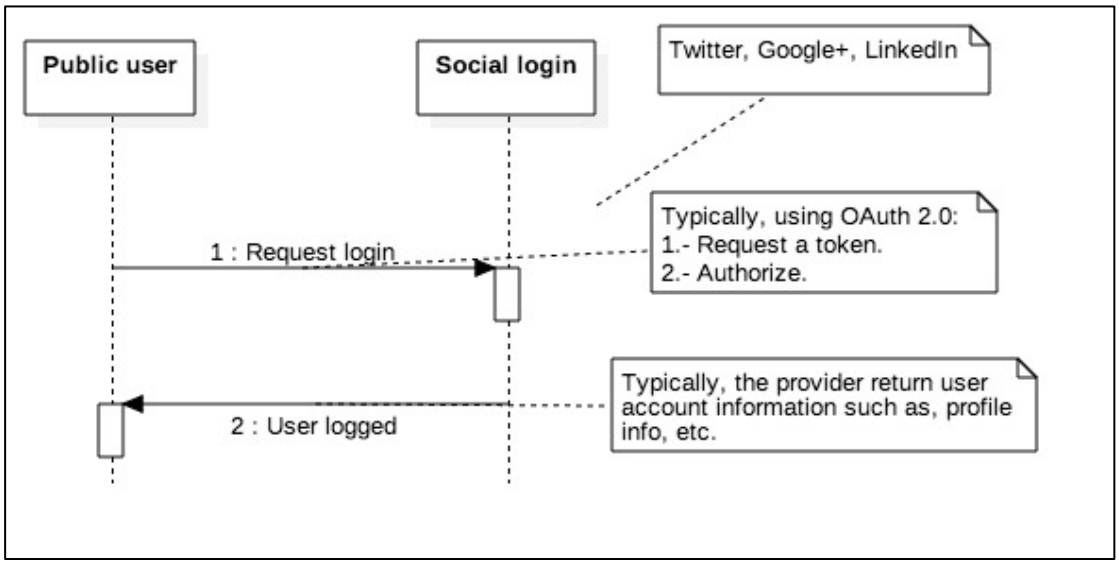


Figure 5: Public user login sequence diagram

Flight search

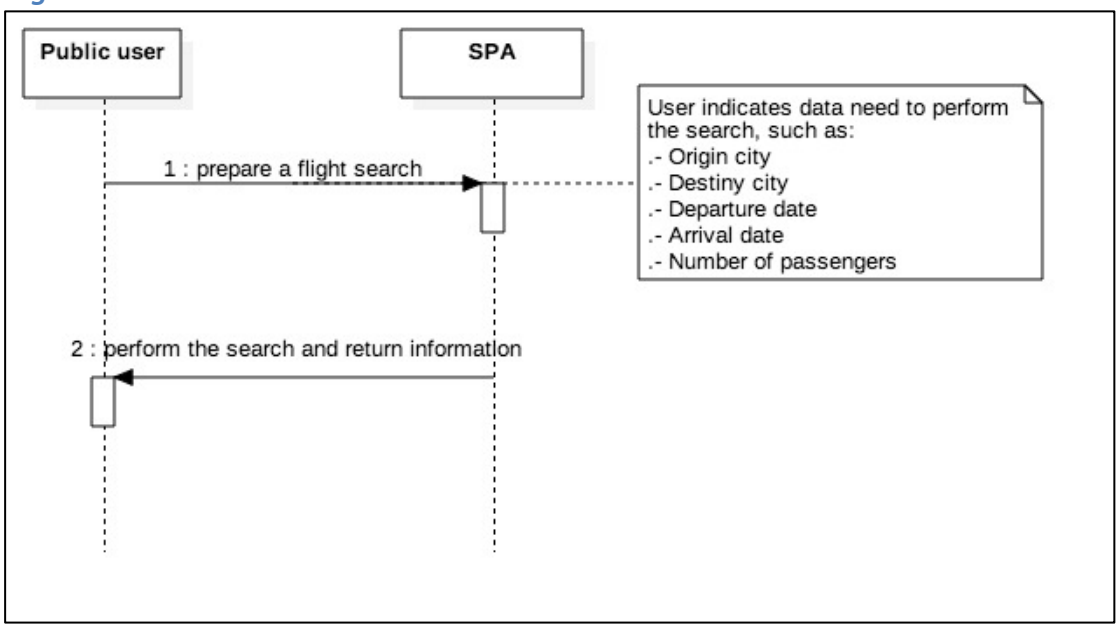


Figure 6: Public user flight search sequence diagram

Booking creation

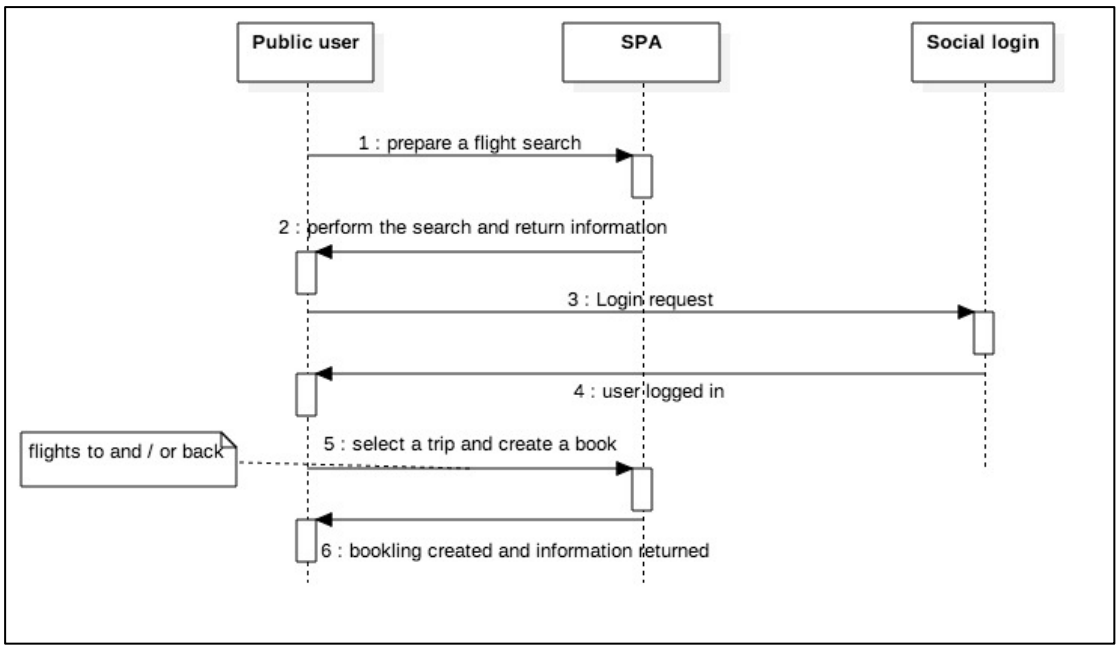


Figure 7: Public user booking creation sequence diagram

Booking cancellation

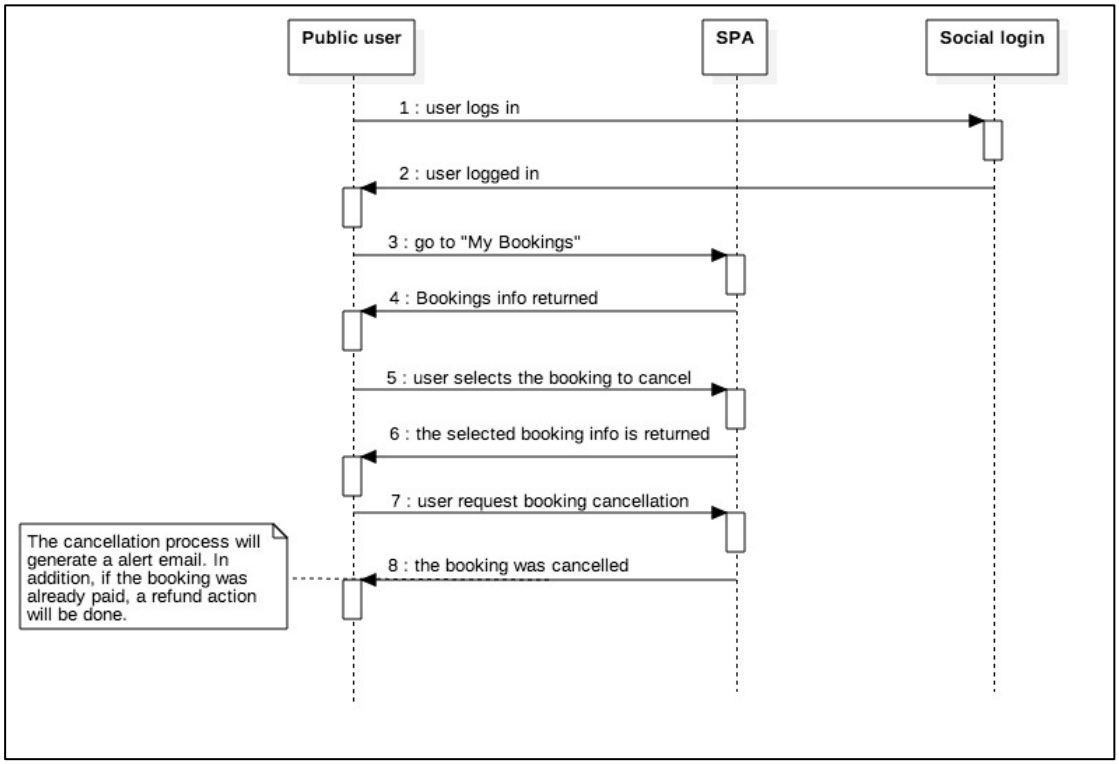


Figure 8: Public user booking cancellation sequence diagram

Booking payment

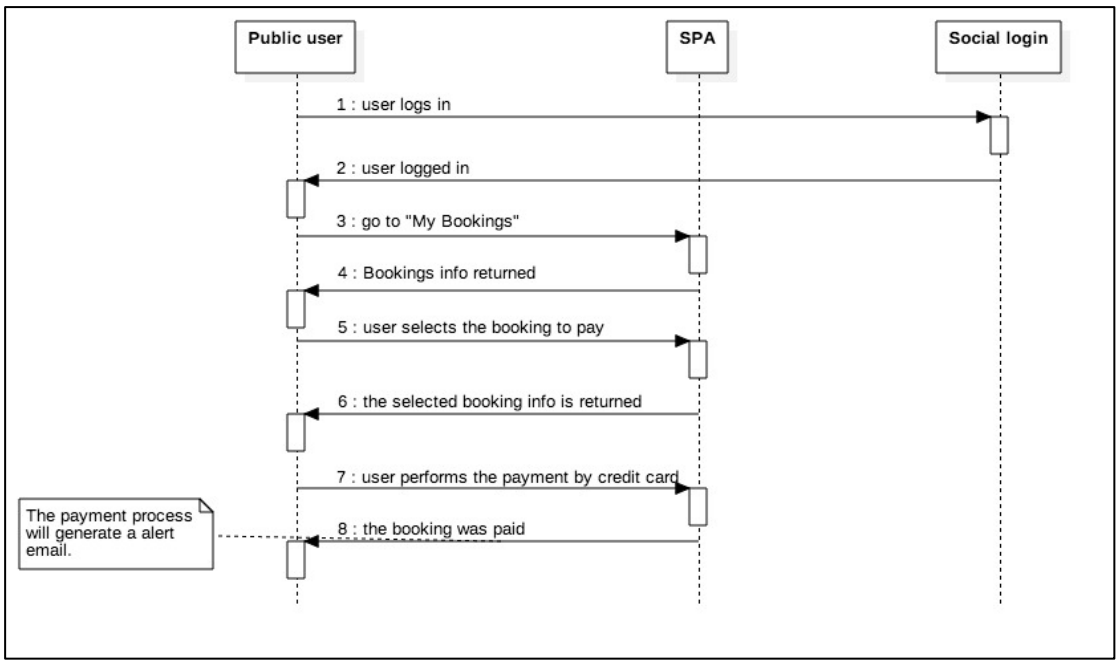


Figure 9: Public user booking payment sequence diagram

Booking check out

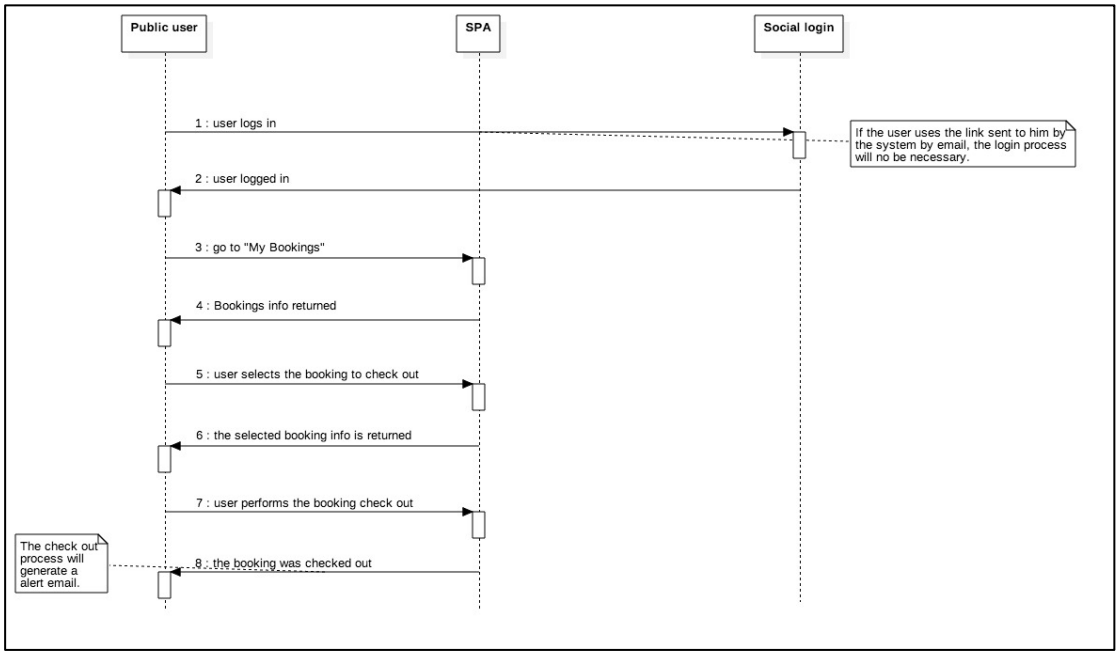


Figure 10: Public use booking check out sequence diagram

Airline user use cases

Login

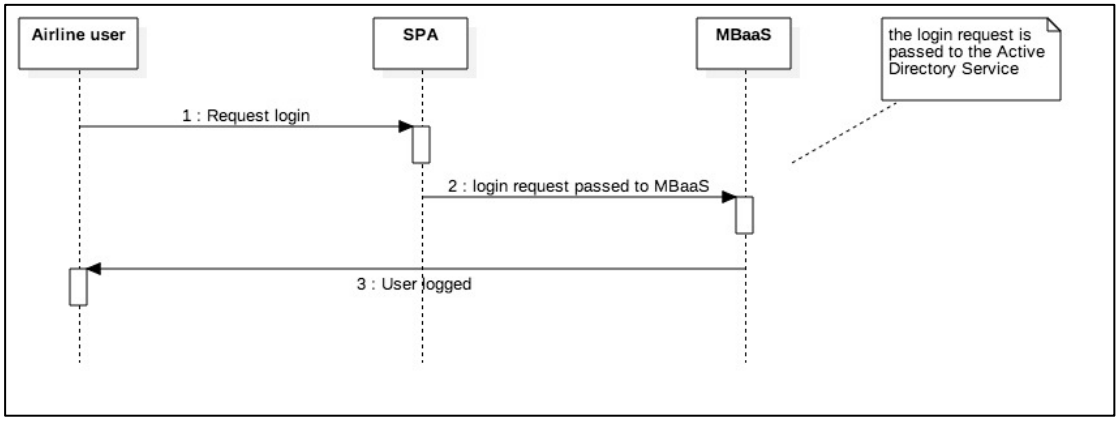


Figure 11: Airline user login sequence diagram

Reservation chart generation

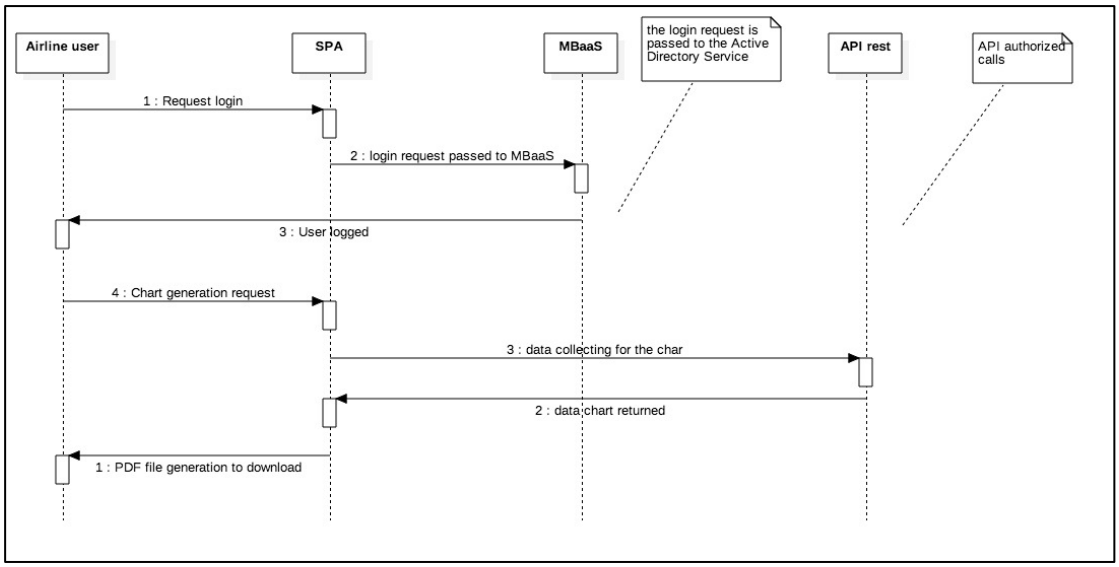


Figure 12: Airline user reservation chart generation sequence diagram

Ticket cancellation

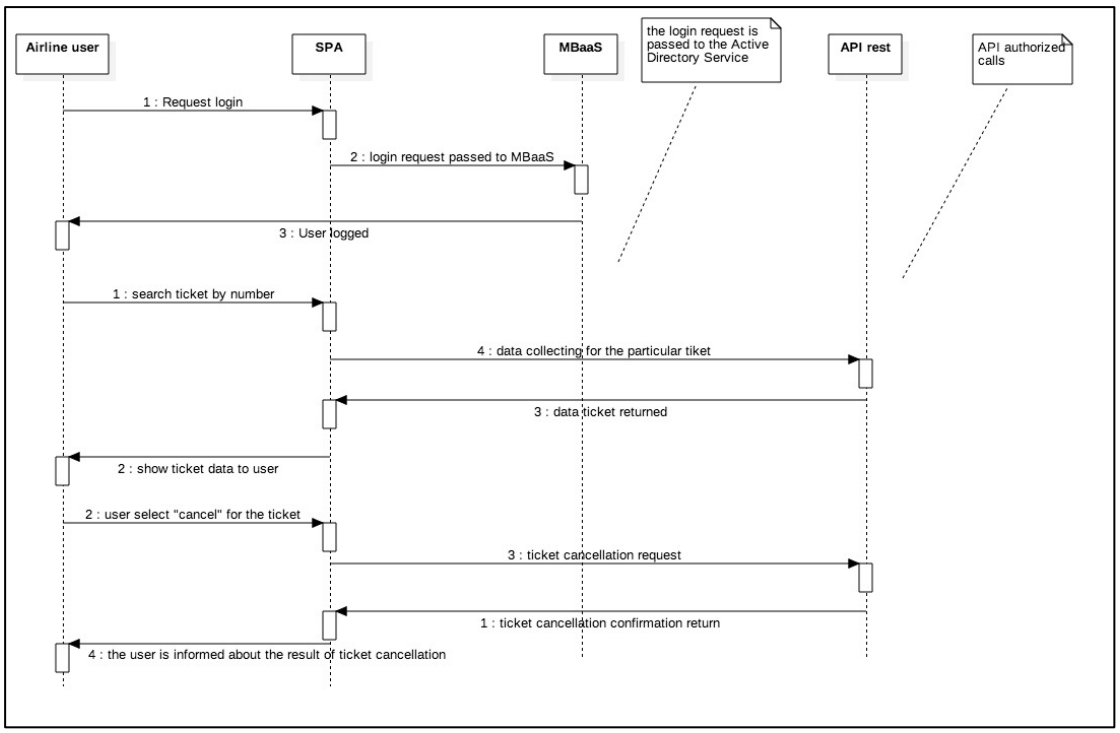


Figure 13: Airline user ticket cancellation sequence diagram

Flight cancellation

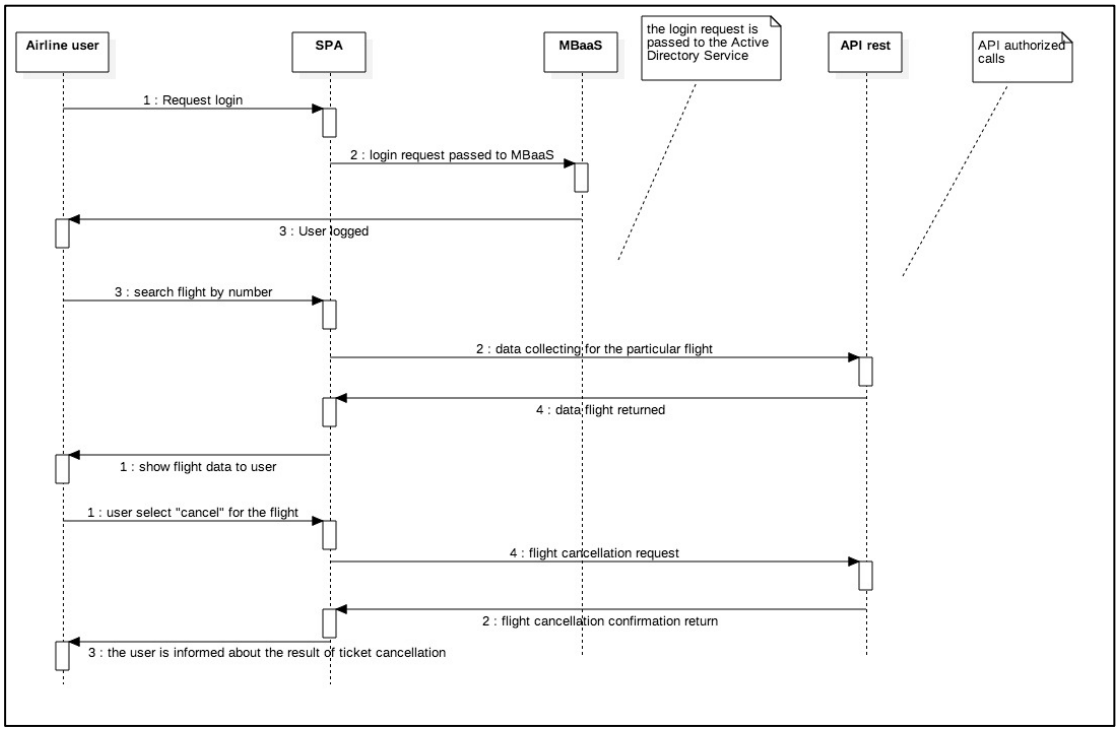
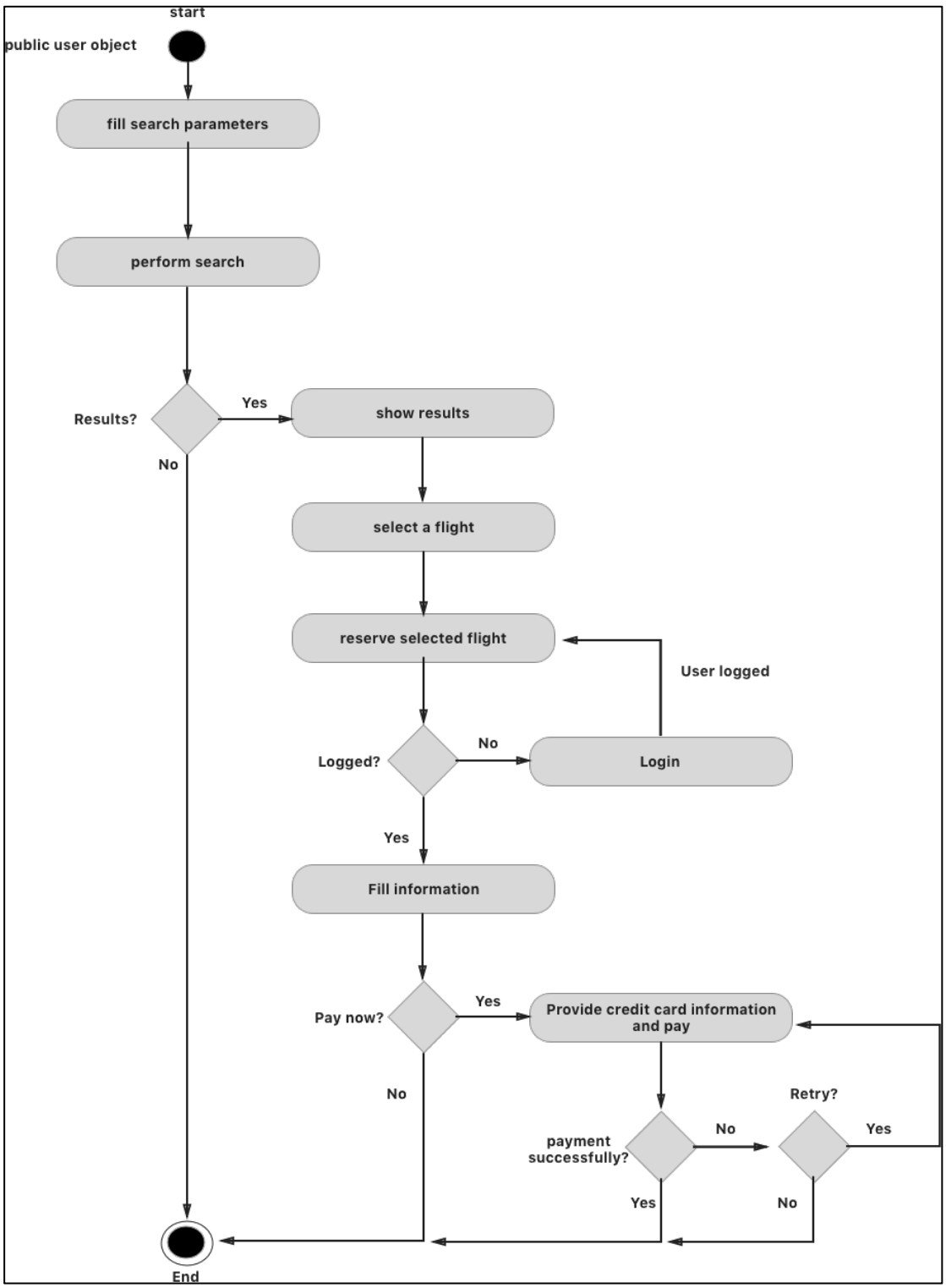


Figure 14: Airline user flight cancellation sequence diagram

Note: Only in the last sequence diagrams was included the MBaaS and de API Rest elements. In the previous they were omitted just to simplify.

Flow diagrams

Following, it shows the flow diagram for search, booking and payment of a flight.



UI and user interaction

This section contains a simple set of UI screenshots to demonstrate the user interaction. The

depicted below show transitions between pages for the main of the process in the software: the search, booking and payment of a flight.

Figure 15: Search, booking and payment of flight - UI prototype

Other comments

In this document, there are some aspects that were omitted. For example:

- Classes diagram
- Activity diagram
- Component diagram
- Physical system design

The main reason is that this document was created as result of a test for software architects during a job selection process. The time needed to perform all diagrams and the rest of omitted information would require a time effort bigger than 48 hours (the time given to complete the test).
On the other hand, sections like "**Flow diagram**" only includes a flow diagram for one of the use cases, just for a demonstration. The same for section "**UI screenshots**"

Is important to say, that some parts of the software architecture and the system that will support it, makes necessary to know more information about the project to develop, for example, the budget available.

In the section "***Other technical and Non-functional requirements***", the point number 5 says: "***technologies are not recommended as it is a design you should make as per the position applied for***". And in section "***What we will evaluate?***" the point number 4 says: "***Technology choices made, design patters applied and the rationale give "***. In my opinion, this is a contradiction. Anyway, I will give information to answer these questions. Because of job position is Chief Software Architect .NET, my proposal will be related to Microsoft tools and technologies.

Technologies proposed:

- **Middleware mobile / web (MBaaS):** Microsoft Azure.
- **API Rest**: Microsoft .NET MVC / WebAPI
- **SOA Service Catalog**: Microsoft .NET framework 4.5
- **Data storage**: Microsoft SQL Server 2012/2014/2016.

Architecture design patters:

- MVC (model view controller) / MVVM (model view view model).
- SOA (Services oriented architecture).

Figure 1: Air Ticket Reservation System Architecture 4
Figure 2: Data model diagram 6
Figure 3: Actors and relations diagram 7
Figure 4: Use cases 8
Figure 5: Public user login sequence diagram 9
Figure 6: Public user flight search sequence diagram 9
Figure 7: Public user booking creation sequence diagram 10
Figure 8: Public user booking cancellation sequence diagram 10
Figure 9: Public user booking payment sequence diagram 11
Figure 10: Public use booking check out sequence diagram 11
Figure 11: Airline user login sequence diagram 12
Figure 12: Airline user reservation chart generation sequence diagram 12
Figure 13: Airline user ticket cancellation sequence diagram 13
Figure 14: Airline user flight cancellation sequence diagram 13
Figure 15: Search, booking and payment of flight - UI prototype 15