

Flight Planning System

———— Project team 6 ————

Project Leader:

Aslam Senouci-Bereksi

Ahmed Guettou, Amine Belkhechine, Helmi Zegaya, Jordan Capaldi, Karthik Sampath,
Lyes Hadjed, Mahmoudou Tallibe Diallo, Mehdi Necibi, Raphael Trancoso, Timera Ibrahima

System Presentation

The system permits to schedule flights (assign a flight crew, join documentation to the flight ...)

The flight system can be edited by different ways (manual modification, automatic ..).

The actors who deal with the system are :

- Members of the flight crew
- OCC members.

Different ways are available for data consulting (mobile interface for the flight crew and desktop interface for OCC members).

A mail alert system is also present to notify information about a flight.

About the Actors

☐ The different system users are :

- ☐ - Members of the flight crew
- ☐ - OCC members

☐ Differences in use between the users :

☐ For flight crew :

- ☐ Mobile interface
- ☐ Only data consulting

☐ For OCC members :

- ☐ Desktop interface
- ☐ All rights (editing / reading / deleting)

Business Objects

Presented with the following form : ***Object (attributes)***

- ☐ **Crew** (Pilot, Co-pilot, Hostess/Steward)

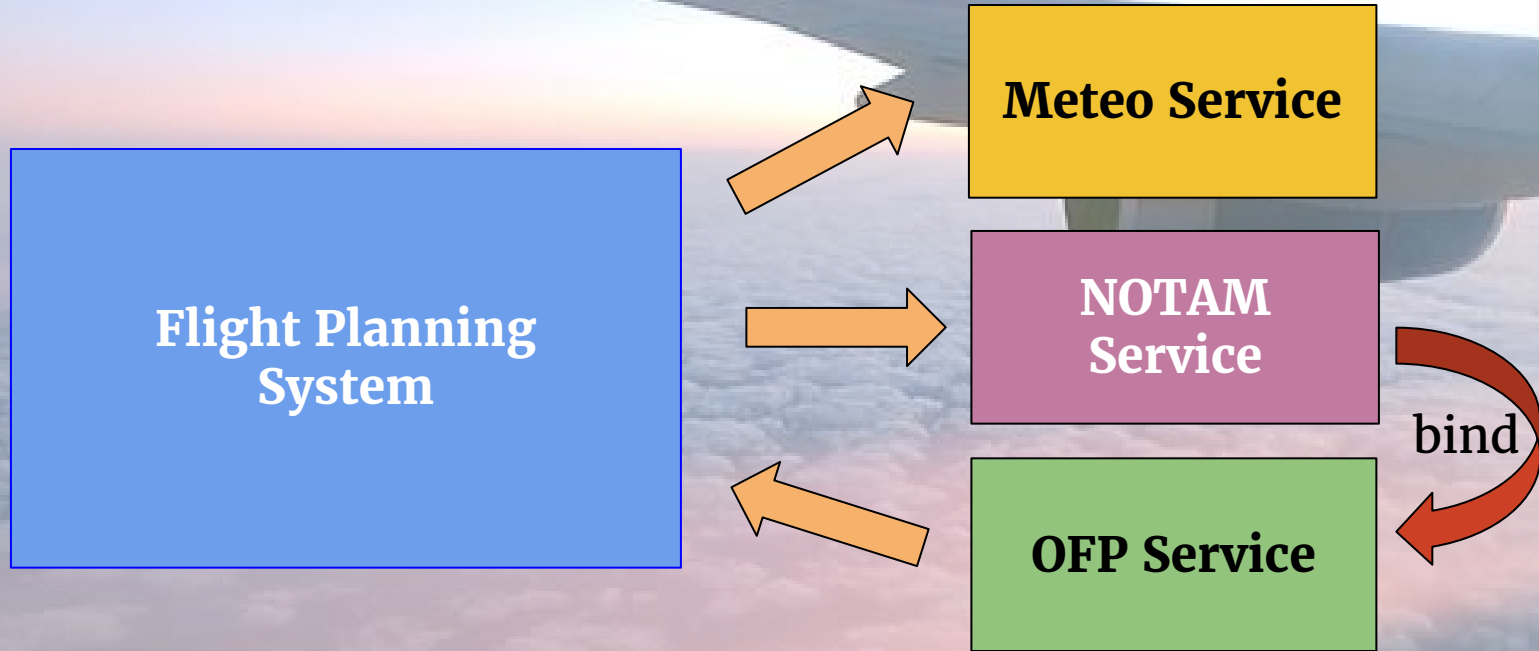
- ☐ **Airplane** (Id, Type, Weight, Size, Capacity)

- ☐ **Airport** (OACI, Name)

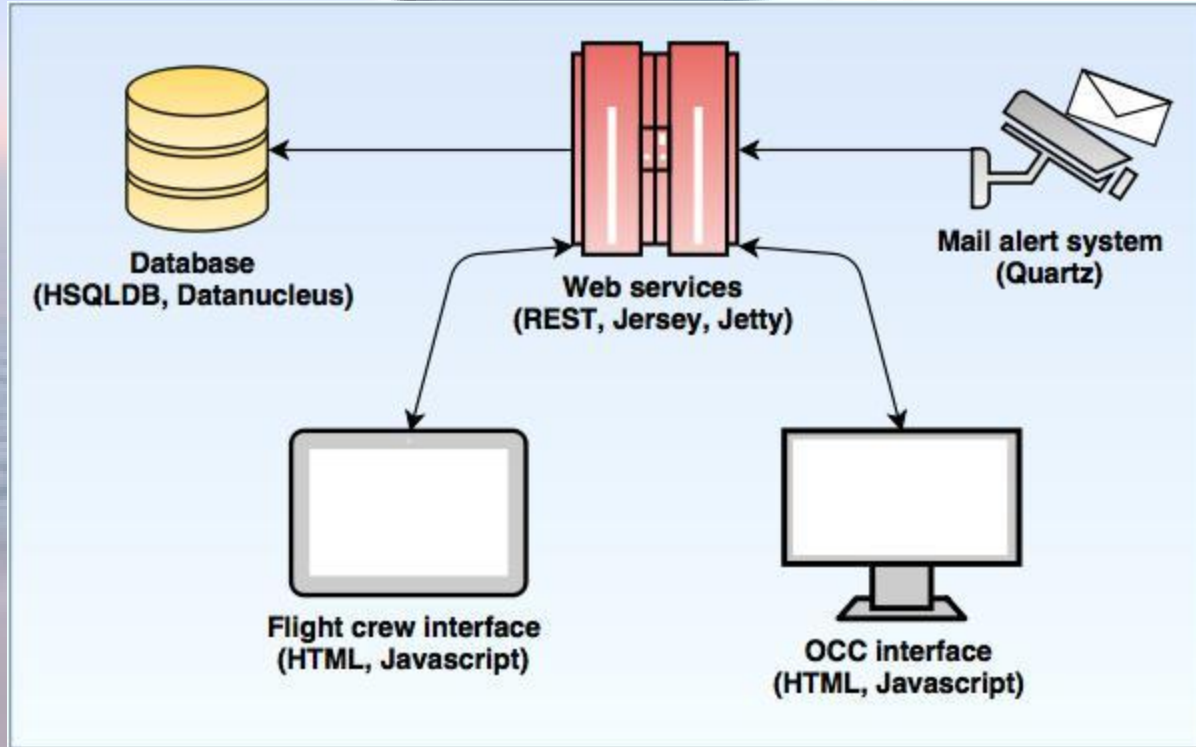
- ☐ **Person** (Login, Type, Name, Surname, Password)

- ☐ **Flight** (Commercial number, ATC number, Arrival date, NOTAM
Departure date, Crew, Arrival OACI, Departure OACI, Meteo, OFP)

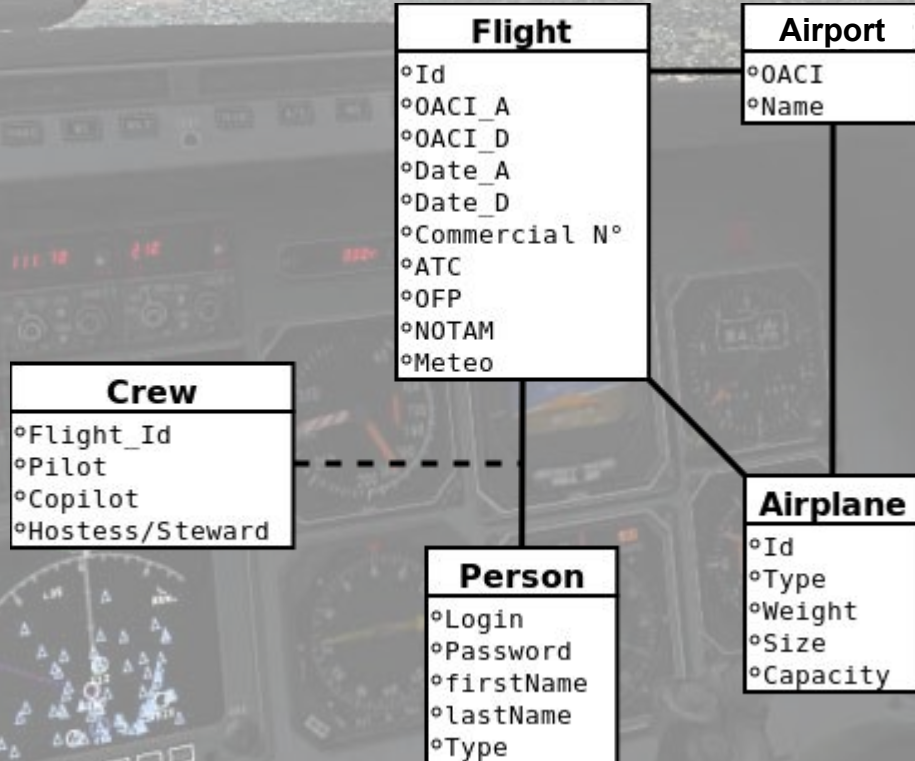
Global System Diagram



Subsystems



Technical Propositions



Task Assignment

Project leader
Aslam

Web-services REST
Jersey, Jetty server

Helmi
(leader)

Aslam

Amine

Database HSQLDB
DataNucleus

Mehdi
(leader)

Lyes

Graphical Interface

Jordan (leader)

OCC Interface

Mahmoudou

Karthik

Flight crew Interface

Timera

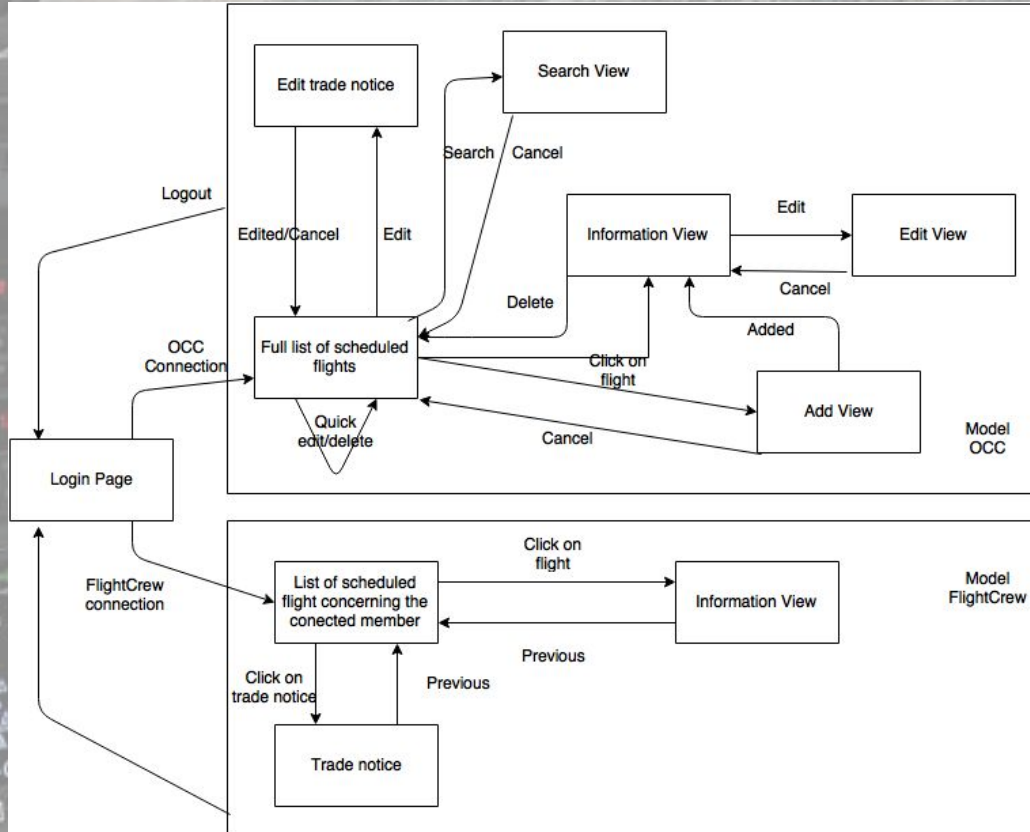
Raphael

Alert system
Quartz

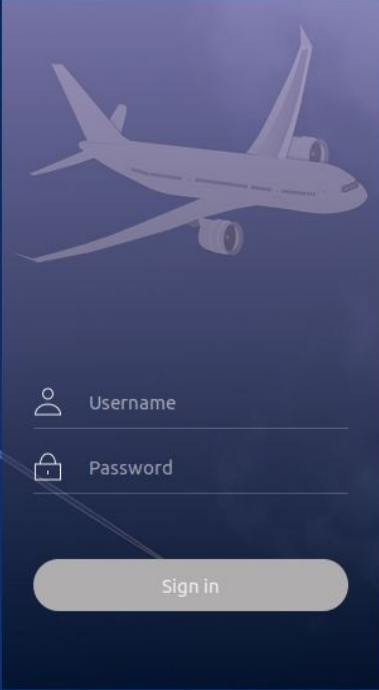
Ahmed
(leader)

Raphael

Navigation Diagram



Mock-ups



A sign in form mock-up on a dark blue background with a faint airplane silhouette. The form is white and contains two input fields: 'Username' with a person icon and 'Password' with a lock icon. A 'Sign in' button is at the bottom.

Username

Password

Sign in

Add View

Home Add Search Log out

Commercial Number :

ATC Number :

Departure Airport :

Arrival Airport :

Planned Departure :

Planned Arrival :

Airplane :

Pilot :

Copilot :

Stewarts/Hostesses :

CANCEL ADD

Mock-ups

Edit View

Home Add Search Log out

Commercial Number :

ATC Number :

Departure Airport :

Arrival Airport :

Planned Departure :

Arrival Departure :

Airplane :

Choose an airplane :

Pilot :

Choose a pilot :

Copilot :

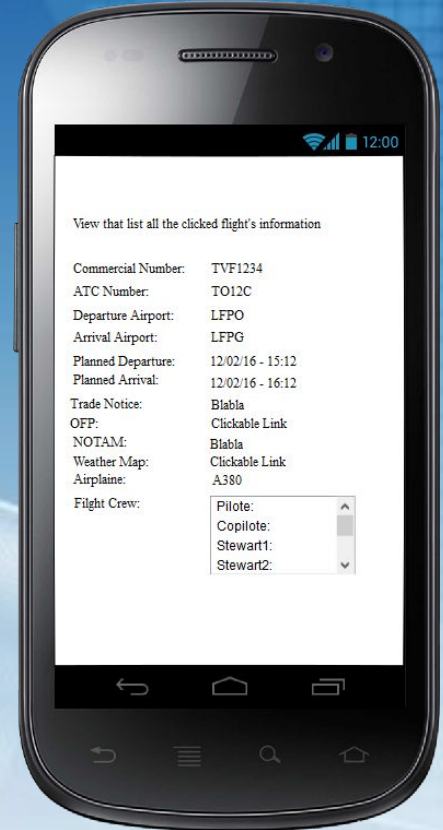
Choose a copilote :

Stewarts/Hostesses :

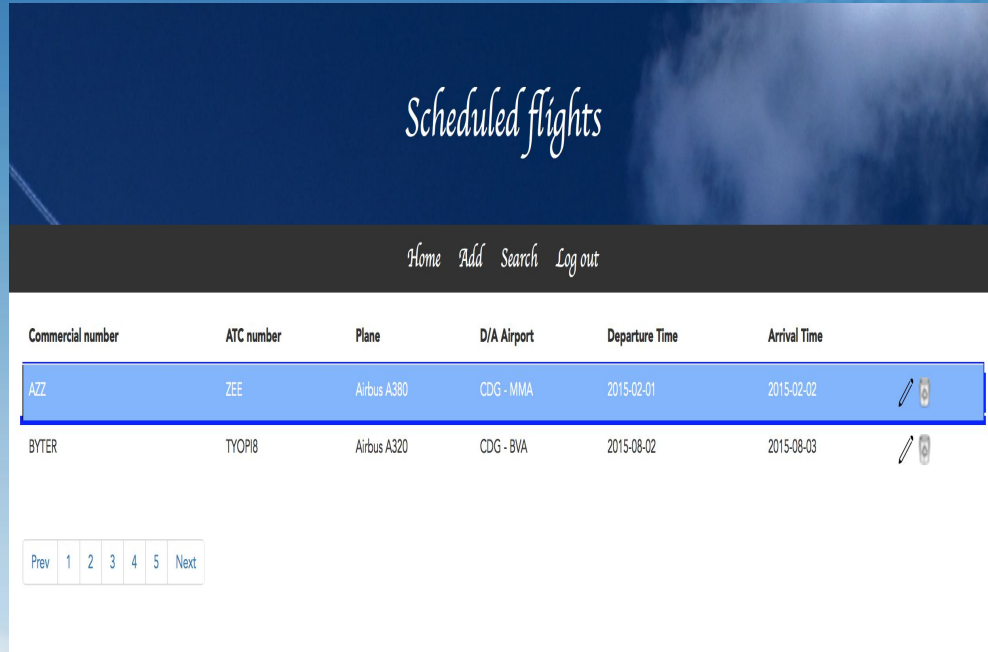
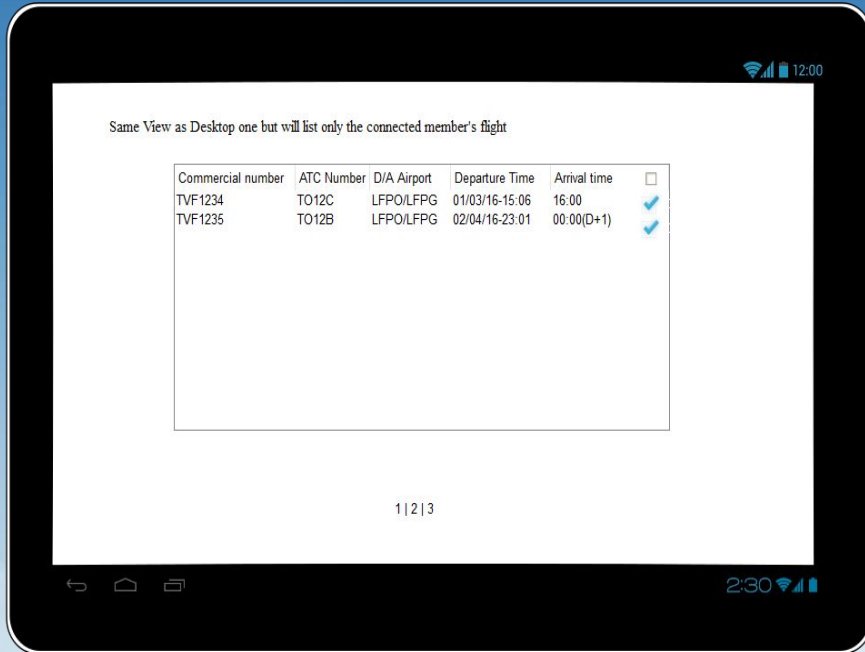
Choose a stewart/hostess :

CANCEL

EDIT



Mock-ups



Mock-ups

Information View

[Home](#) [Add](#) [Search](#) [Log out](#)

Commercial Number :	BYTER	OFF :	ofp.pdf
ATC Number :	TYOPI8	NOTAM :	notam.txt
Departure Airport :	CDG	Weather Map :	meteo.jpg
Arrival Airport :	BVA	Airplane :	Airbus A320
Planned Departure :	2015-08-02	Flight Crew :	
Planned Arrival :	2015-08-03		

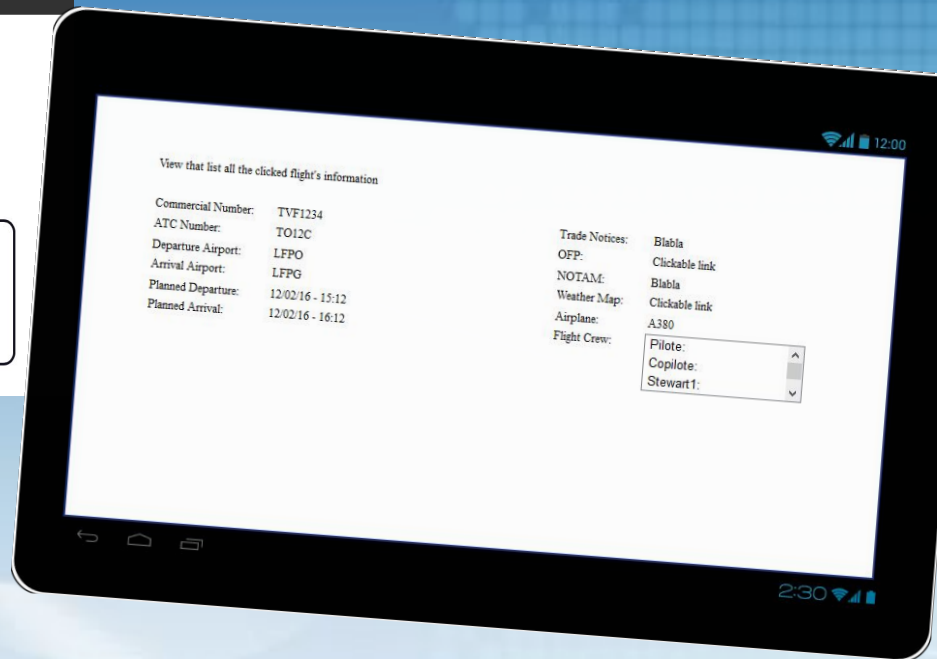
Pilot : Aslam
Co-Pilot : Lyes
Stewards/Hostesses : Katie

View that list all the clicked flight's information

Commercial Number:	TVF1234
ATC Number:	TO12C
Departure Airport:	LFPO
Arrival Airport:	LFPG
Planned Departure:	12/02/16 - 15:12
Planned Arrival:	12/02/16 - 16:12

Trade Notices:	Blabla
OFF:	Clickable link
NOTAM:	Blabla
Weather Map:	Clickable link
Airplane:	A380
Flight Crew:	

Pilote:	^
Copilote:	
Stewart1:	v



JavaScript

```
function callGetDone(result) {  
    var templateExample = _.template($('#template').html());  
    var flightList = $.parseJSON(JSON.stringify(result));  
    sessionStorage.myValue = JSON.stringify(flightList[0].atc);  
    for (var i = 0; i < flightList.length; i++) {  
        getAirplaneType(flightList[i].idAirplane);  
        var html = templateExample({  
            "commercialNumber": flightList[i].commercialNumber,  
            "atcNumber": flightList[i].atc,  
            "airplane": airplaneType,  
            "oaciDeparture": flightList[i].oaciDeparture,  
            "oaciDestination": flightList[i].oaciDestination,  
            "departure": flightList[i].departure,  
            "arrivalTime": flightList[i].arrivalTime  
        });  
        $('#newLine').append(html);  
    }  
}
```

```
<script id="template" type="text/template">  
<tr class="clickableRow" data-href="url://info.html">  
    <td>  
        <%= commercialNumber %>  
    </td>  
    <td>  
        <%= atcNumber %>  
    </td>  
    <td>  
        <%= airplane %>  
    </td>  
    <td>  
        <%= oaciDeparture %> - <%= oaciDestination %>  
    </td>  
    <td>  
        <%= departure %>  
    </td>  
    <td>  
        <%= arrivalTime %>  
    </td>  
</tr>  
</script>
```

JavaScript

```
$.ajax({
  contentType: "application/json",
  url: url,
  data: JSON.stringify({
    "departure": departure, "oaciDeparture": oaciDep,
    "commercialNumber": commNum, "oaciDestination": oaciArr,
    "arrivalTime": arrival, "atc": atc, "ofp": "ofp.pdf",
    "notam": "notam.txt", "meteo": "meteo.jpg", "tradeNotice": tradeNot,
    "crew":
    { "loginPilot": pilot, "loginCopilot": copilot,
      "loginHostStaff": staffVal }, "idAirplane": plane }),
  type: "PUT",
  processData: false
}).done(function() {
  alert( "The flight has been added." );
  // Cleaning of input fields here
}).fail(function() {
  alert("An error has occurred. Please check informations.");
});
```


Web-services

METHOD	URL	BEHAVIOUR
GET	/ws/flights/{id}	Returns the flight information of the corresponding id
POST	/ws/flights/{id}	Modify an existing flight for the corresponding id
POST	/ws/flights	Returns the list of flights
PUT	/ws/flights	Add a new flight
DELETE	/ws/flights/{id}	Delete an existing flight for the corresponding id
POST	/ws/	Connect the user to the system
GET	/ws/pilot	Returns the whole list of pilots
GET	/ws/copilot	Returns the whole list of copilots
GET	/ws/hoststaff	Returns the whole list of hostess and steward
GET	/ws/airplanes	Returns the whole list of airplanes
GET	/ws/flights/{id}/ofp	Returns the ofp of the flight {id}
GET	/ws/flights/{id}/notam	Returns the NOTAM of the flight {id}
GET	/ws/flights/trade	Returns the trade notice (pdf)
POST	/ws/flights/trade	Modify an existing trade notice
GET	/ws/airports	Returns the whole list of airports

Jersey Implementation

```
@GET
@Produces(MediaType.APPLICATION_JSON)
@Path("/hoststaff")
public List<Person> getHostStaff() {
    return ((List<Person>)pdi.
getHostStaff());
}
```

```
@POST
@Produces(MediaType.APPLICATION_JSON)
@Path("/flights")
public List<Flight> getFlights() {
    return ((List<Flight>)fdi.getFlights());
}
```

```
@POST
@Consumes(MediaType.APPLICATION_JSON)
@Produces(MediaType.APPLICATION_JSON)
public Response authenticate(Person person) throws
WebApplicationException{

    HttpSession session = req.getSession(true);
    // Set timeout value in second
    session.setMaxInactiveInterval(20*60);
    Person retrieved = pdi.checkUser(person.getLogin(),
    person.getPassword());
    if (retrieved == null)
        throw new WebApplicationException(
            Response.status(
                Response.Status.NOT_ACCEPTABLE).build());
    session.setAttribute("username", retrieved.getLogin());
    return Response.status(Response.Status.ACCEPTED).build();
}
```


Java interfaces

```
public interface PersonDao {
```

```
/**
```

```
 * @return the list of pilots
```

```
 */
```

```
List<Person> getPilots();
```

```
/**
```

```
 * @return the list of co-pilots
```

```
 */
```

```
List<Person> getCoPilots();
```

```
/**
```

```
 * @return the list of staff
```

```
 */
```

```
List<Person> getHostStaff();
```

```
/**
```

```
 * @param the user who wants to log in
```

```
 */
```

```
Response authenticate(Person person);
```

```
public interface AirplaneDao {
```

```
/**
```

```
 * @return the list of all available airplanes
```

```
 */
```

```
List<Airplane> getAirplanes();
```

```
}
```


Java interfaces

```
public interface FlightDao {
```

```
/**
```

```
 * @return this list of Flight
```

```
 */
```

```
List<Flight> getFlights();
```

```
/**
```

```
 * @param departure
```

```
 * @return the list of Flights assigned to  
        a specific departure date.
```

```
 */
```

```
Flight getFlight(String id);
```

```
/**
```

```
 * @return a flight crew
```

```
 * @param departure
```

```
 */
```

```
Crew getCrew(String id);
```

```
...
```

```
/**
```

```
 * @param departure
```

```
 * @return The URI of the OFP
```

```
 */
```

```
String getOFP(String id);
```

```
/**
```

```
 * @param departure
```

```
 * @return The URI of the NOTAM
```

```
 */
```

```
String getNOTAM(String id);
```

```
/**
```

```
 * @return The URI of the trade notice
```

```
 */
```

```
String getTradeNotice();
```

```
...
```

```
}
```