AirBooking. Reserves en línia

Memòria del projecte

Treball de Fi de Carrera

Josep Manel Martorana

Consultor: Joan – Vicent Orenga

Gener 2008, Barcelona

1. RESUM DEL TREBALL DE FI DE CARRERA

L'objectiu d'aquest treball de fi de carrera és el doble:

1. desenvolupar un projecte de desenvolupament, aplicant tots els coneixements adquirits durant la carrera i
2. investigar les darreres tecnologies al mercat en aplicacions empresarials

Per tal d'assolir el primer objectiu, he decidit implantar una lloc web de reserves de vol en línia. He cobert totes les fases del seu cicle de vida (requeriments, anàlisi, disseny, construcció i desplegament). L'anàlisi i el disseny s'han documentat formalment mitjançant UML i la construcció s'ha efectuat sobre la plataforma de desenvolupament JavaEE 5.

De cara a l'aprofundiment en les tecnologies més innovadores en aplicacions empresarials, he escollit EJB 3, JSF i JPA, sense tenir cap coneixement inicial de cap d'elles.

2. AGRAÏMENTS

M'agradaria agraït a tot l'equip docent de la UOC el posar uns estudis universitaris que considero d'alta qualitat a l'abast de persones que ja estem immersos en el món laboral i que, altrament, en seria impossible accedir a uns de similars.

ÍNDEX

3. INTRODUCCIÓ

3.1 Justificació del TFC: punt de partida i aportació

El punt de partida d'aquest TFC era doble: d'una banda els coneixement d'enginyeria del programari i programació obtinguts durant la carrera i, de l'altra, l'experiència professional després de 5 anys en el mont del desenvolupament de programari.

Així doncs, no es pretenia tant desenvolupar una aplicació que resolgués un problema del món real, si no estudiar i experimentar amb algunes de les darreres tecnologies existents al mercat pel desenvolupament d'aplicacions empresarials, i veure com aquestes milloren la productivitat dels projectes, així com la seva robustesa. En conclusió, no s'ha perseguit tant una solució funcionalment i tècnica correcta com la inclusió de mecanismes tèccics innovadors i interessants.

Amb aquest objectiu, a banda de la orientació a objectes i dels patrons descrits al capítol de disseny tècnic (MVC, *front controller* i DAO) he fet servir les següents tecnologies:

- **JDK 5:** el salt entre la versió 1.4 i 1.5 del llenguatge va suposar una sèrie de canvis molt importants, tant com per canviar la versió del mateix (de 1.x a 5). En el projecte he aprofundit especialment en la programació genèrica, tal i com s'explica en el capítol dedicat a aquest punt, s'ha aconseguit, entre altres, crear una pantalla de manteniment única per totes les dades mestres gràcies a aquesta característica.
- **EJB 3:** no només s'ha utilitzat la versió 5 del llenguatge, sinó que tot el projecte està basat en Java EE 5. EJB 3 és la nova especificació dels Java Enterprise Beans, molt més lleugera i senzilla d'utilitzar que la 2. Algunes de les característiques que incorporen són el suport per anotacions o la inversió de control mitjançant la injecció de dependències. A més de per implementar la lògica de negoci amb *Session Beans*, s'ha utilitzat també MDB's i *entity beans* per la persistència.
- **JPA:** *Java Persistence Api* és l'api definida per Sun Microsistems per tal d'estandarditzar la persistència dels sistemes. JPA és només una especificació i cal triar una implementació concreta, al projecte s'ha utilitzat amb aquest fi, triant TopLink Essentials com a implementació.
- **MDB:** els *Message Driven Beans* són la manera més senzilla i potent d'implementar asincronismes en un sistema mitjançant *Java Message Service* (JMS). A projecte he implementat un per demostrar com pot ser una solució vàlida en casos en que existeix una operació de temps indeterminat i es vol donar una resposta a l'usuari de forma immediata.
- **JSF:** *Java Server Faces* és el *framework* per la capa de presentació en aplicacions web proposat per Sun. Es caracteritza per la seva orientació a components i pel seu

model d'esdeveniments en la banda del servidor. JSF és només una especificació, pel projecte s'ha triat Richfaces, de RedHat, sobre la implementació MyFaces d'Apache.

- **RichFaces:** es tracta d'un projecte de Red Hat que consisteix en una sèrie de components avançats (arbres, menús, calendaris, etc.) que complementa qualsevol implementació dels components bàsics definits a JSF (en el projecte s'ha emprat MyFaces). La característica principal d'aquest projecte és la inclusió de tècniques AJAX (*Asynchronous Javasctipt and XML*), permetent actualitzar seccions de la plana web sense la necessitat de recarregar-la, i sense la necessitat de que el desenvolupador conegui javascript.

Cal destacar que, tot i que sí tenia experiència en desenvolupaments JavaEE abans d'aquest TFC (Struts, Spring, jdbc, etc.), mai no havia emprat cap de les tecnologies enumerades, fet que ha implicat un esforç important d'investigació, però també a suposat un bon aprofitament del TFC amb coneixements pràctics.

3.2 Objectius del TFC

Tal i com s'ha explicat abans, el principal objectiu d'aquest TFC ha estat aprofundir en el coneixement d'algunes de les tecnologies més innovadores en el desenvolupament d'aplicacions empresarials dins el món JavaEE.

A més d'emprar patrons com MVC o DAO, s'han emprat les tecnologies descrites anteriorment (JavaEE 5, JPA, MDB, JSF, Richfaces, ...).

En segon terme, també s'ha pretès seguir el cicle de vida normal pel desenvolupament d'aquesta mena de projectes, realitzant un anàlisis i disseny tècnic formals.

3.3 Enfocament i mètode seguit

L'enfocament i el mètode per encarar el projecte ha estat d'una banda, el del cicle de vida clàssic (anàlisi, disseny, construcció), seguint un paradigma orientat a objectes i generant una documentació formal mitjançant UML, i de l'altra, la investigació i aprenentatge de les diferents tecnologies que s'han emprat al projecte.

Sense cap mena de dubte, la principal dificultat del projecte (tot i ser també el seu principal objectiu) ha estat l'aprenentatge d'aquestes tecnologies en tan poc temps. Un projecte del "món real" acostuma durar més temps i a existir algú dins l'equip que té alguna mena de coneixement sobre les tecnologies emprades. Haver d'investigar totes aquestes tecnologies ha estat un gran repte i, alhora, una gran satisfacció.

3.4 Planificació del projecte

Bàsicament crec que la planificació inicial del projecte s'ha respectat. Tot i que no es van assolir totes les fites marcades per a la PAC 3, els punts que restaven d'implementar eren mínims i els he pogut finalitzar amb l'entrega final.

Aquesta és la planificació inicial:

- 29/10/2007: Entrega PAC 2, que inclourà
  - Anàlisi funcional de l'aplicació
  - Disseny tècnic de l'aplicació
- 17/12/2007: Prototipus operatiu, inclourà les funcions següents:
  - Reserves de vols
  - Autenticació
  - Consulta de Reserva
- 14/01/2008: Entrega final.
  - Tot el projecte i la documentació
  - Memòria final.

3.5 Productes obtinguts

El principal producte obtingut fruit del projecte és una aplicació Java EE de reserves de vols en línia. Es tracta d'una aplicació web distribuïda composada per un módul war (web app) i un jar (ejb). També s'inclou tota la documentació funcional i tècnica, així com el javadoc de les principals classes.

3.6 Descripció de la resta de capítols de la memòria

A la resta de capítols es detalla la presa de requeriments, l'anàlisi funcional i tècnic del projecte i finalment, la guia d'instal·lació.

4. REQUERIMENTS FUNCIONALS

A continuació es descriuen tots els requeriments funcionals que el sistema ha de complir.

4.1 Alta d'usuari

Quan un usuari potencial arriba al nou web ha de poder registrar-se al sistema de forma fàcil i ràpida. Amb aquest objectiu, el sistema li oferirà un formulari on haurà d'informar de les dades personals necessàries per poder efectuar les reserves.

Un cop registrat, l'usuari ja podrà fer un ús normal del sistema.

4.2 Autenticació

Un cop registrats al sistema, els usuaris hauran d'autenticar-se per poder fer-ne ús. La identificació es farà mitjançant nom d'usuari i paraula de pas.

4.3 Reserves de vols

Ha de ser la pantalla principal de l'aplicació i haurà de permetre a un usuari autenticat al sistema reserva un vol.

Les funcionalitats que ha d'oferir aquesta opció seran les següents:

- Cercar l'aeroport origen
- Cercar l'aeroport destí (mostrant només aquells pels quals existeixi ruta)
- Escollir la data i hora del vol dins de la disponibilitat horària
- Confirmar la reserva

4.4 Consulta de reserves

Un cop l'usuari s'ha autenticat al sistema ha de tenir accés a consultar les seves reserves. Es presentaran totes elles en una taula, mostrant l'hora de sortida i arribada del vol, els aeroports implicats, així com les companyies i les escales corresponents.

Aquesta pantalla també permetrà anul·lar les reserves sempre que es trobin dins un marge de temps de seguretat.

4.5 Manteniment de dades mestre

Es tracta d'un mòdul genèric que ha de permetre mantenir totes les dades mestres de l'aplicació, incloent:

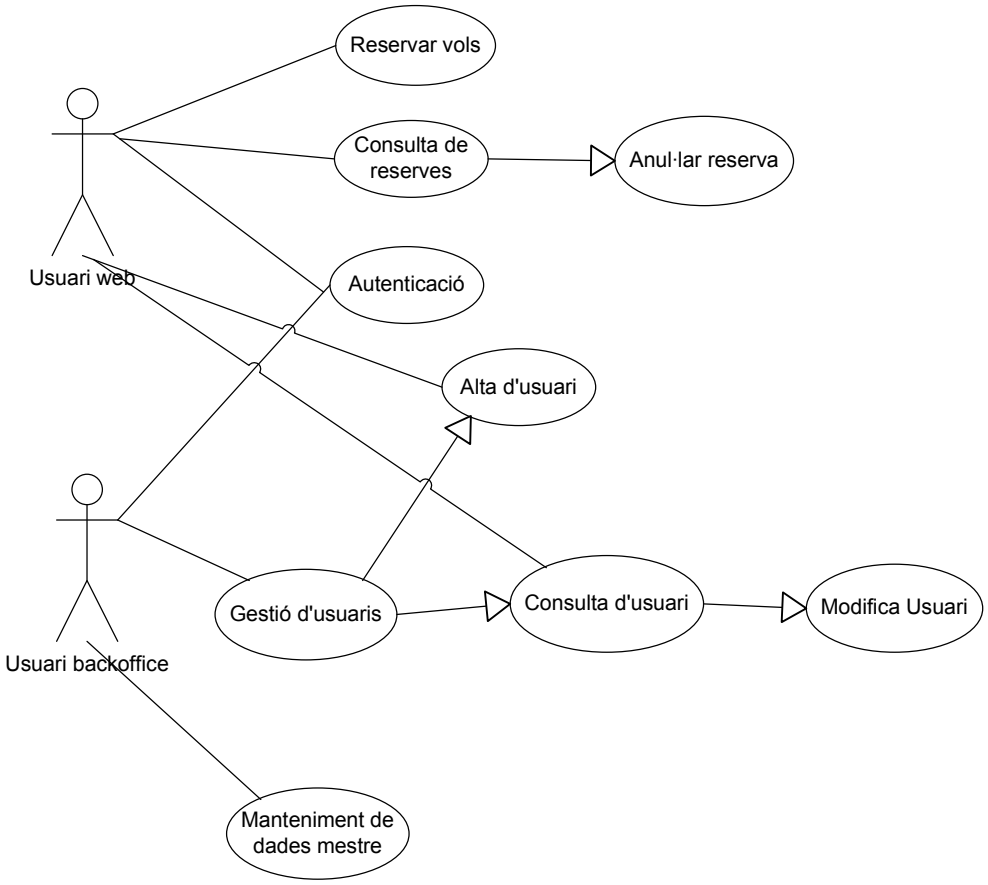
- Aeroports
- Companyies
- Vols
- Etc.

4.6 Gestió d'usuaris

Es tracta del mòdul d'administració de seguretat. Ha de permetre crear usuaris. S'han de poder crear, consultar, donar de baixa i canviar la paraula de pas (en aquest últim cas, la nova paraula de pas s'enviaria per correu electrònic a l'usuari).

5. ANÀLISI FUNCIONAL – CASOS D'ÚS

5.1 Diagrama general de casos d'ús



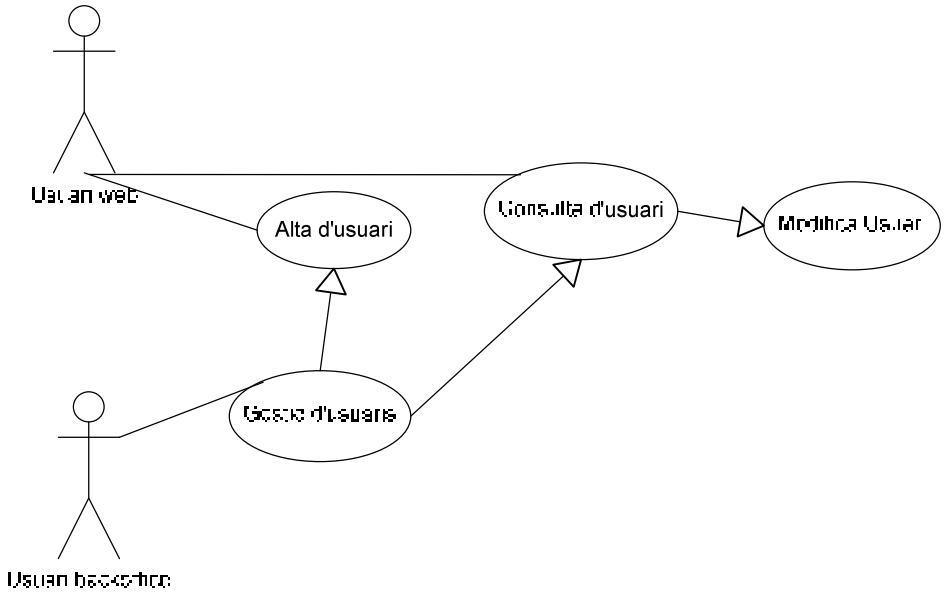
Tal i com es pot veure a la imatge, el sistema compta amb dos actors:

- **Usuari web:** es tracta dels usuaris que accediran des de Internet per fer reserves. Tindran accés als següents casos d'ús:
  - Reservar vols
  - Consulta de reserves
  - Autenticació
  - Alta d'usuari
  - Consulta d'usuari
- **Usuari backoffice:** són els usuaris propis d'AirBooking i que s'encarregaran del manteniment de l'aplicació. Tindran accés als següents casos d'ús:
  - Autenticació

- Gestió d'usuaris
- Manteniment de dades mestre

5.2 Gestió d'usuaris

5.2.1 DIAGRAMA



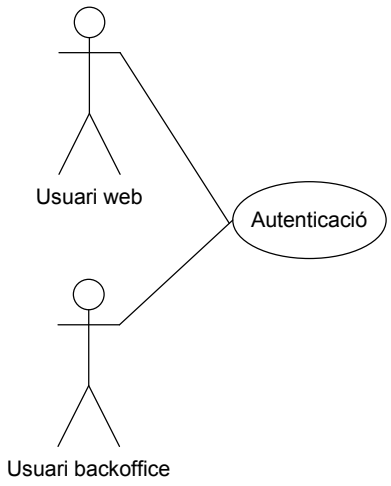
5.2.2 DESCRIPCIÓ

L'objectiu d'aquest cas d'ús es permetre el manteniment i consulta d'usuaris al sistema. L'activitat "Gestió d'usuaris" només estarà disponible pels usuaris backoffice doncs ha de permetre fer cerques sobre el conjunt d'usuaris i escollir sobre quin es vol efectuar l'alta, consulta i modificació.

L'usuari web tindrà la opció d'accedir directament sobre les activitats de consulta i modificació, però únicament sobre el seu usuari. En el cas de l'activitat d'alta, es proporcionarà als usuaris web que encara no estiguin autenticats al sistema.

5.3 Autenticació

5.3.1 DIAGRAMA



5.3.2 DESCRIPCIÓ

Un cop registrats al sistema, els usuaris hauran d'autenticar-se per poder fer-ne ús. La identificació es farà mitjançant nom d'usuari i paraula de pas.

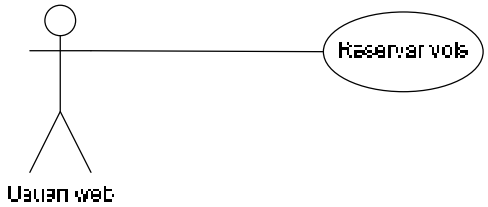
Aquest cas d'ús haurà de permetre, tant als usuaris web com als del backoffice d'identificar-se per poder començar una sessió amb el sistema.

Cal tenir presents els següents punts relacionats amb aquest apartat:

- Les paraules de pas s'emmagatzemaran codificades.
- S'oferirà una opció de recordatori de la paraula de pas.
- S'aplicaran regles de seguretat a les paraules de pas (longitud mínima, combinació majúscules / minúscules, etc.

5.4 Reserves de vols

5.4.1 DIAGRAMA



5.4.2 DESCRIPCIÓ

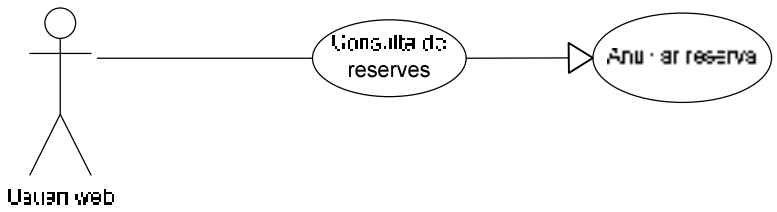
Aquest cas d'ús, només disponible pels usuaris web, ha d'oferir la possibilitar d'efectuar una reserva d'un vol. Amb aquest objectiu, es demanaran les dades necessàries a tal efecte:

- **Aeroport origen:** s'oferirà un control de cerca per poder-ne escollir un d'una llista
- **Aeroport destí:** s'oferirà un control de cerca per poder-ne escollir un d'una llista amb aquells aeroports pels quals hi ha vol des de l'aeroport d'origen escollit
- **Data i hora del vol dins de la disponibilitat horària**: s'oferirà un control de cerca per poder-ne escollir un d'una llista
- **Número de bitllets**

Un cop confirmada la reserva, el sistema realitzarà la sol·licitud al sistema de la companyia corresponent. Amb l'objectiu de millorar l'experiència dels nostres usuaris s'evitarà que el rendiment del sistema depengui del de la companyia on es vol reservar. Per tant, aquesta comunicació haurà de ser asíncrona i es notificarà a l'usuari que s'ha sol·licitat la reserva, en el moment en que el sistema de la companya indiqui que la reserva ha estat satisfactòria o no, es comunicarà aquest esdeveniment a l'usuari mitjançant correu electrònic.

5.5 Consulta de reserves

5.5.1 DIAGRAMA



5.5.2 DESCRIPCIÓ

Només disponible, inicialment, pels usuaris web, permetrà que aquest vegi l'històric de reserves que ha realitzat.

Les consultes es mostraran per ordre cronològic invers (primer les més recents) i de forma paginada. Es presentaran de forma tabular, mostrant l'hora de sortida i arribada del vol, els aeroports implicats, el nombre de bitllets, així com les companyies i l'estat de la reserva.

Els possibles estats seran:

- **Reservat:** s'ha realitzat la reserva, però encara estem a temps per dur a terme la cancel·lació (més d'una setmana abans del vol).
- **Confirmat:** s'ha realitzat la reserva i ja no es pot cancel·lar (menys d'una setmana abans del vol).
- **Cancel·lat:** el vol s'ha cancel·lat.

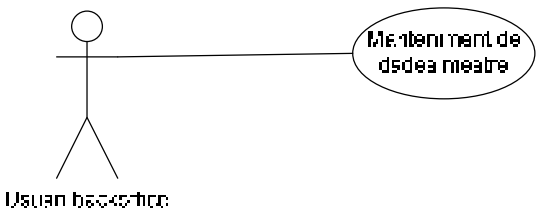
- **Efectuat:** el vol no es va cancel·lar i ja s'ha realitzat.
- **Reserva sol·licitada:** el sistema d'AirBooking ha realitzat la sol·licitud de reserva als sistemes de la companyia que apliqui però aquesta encara no ha comunicat el final del procés.
- **No disponible:** si no el sistema de la companyia no es capaç de realitzar la reserva, aquesta passa a no disponible.

Addicionalment, per cada registre, si aquest es troba en estat reservat, es donarà l'opció mitjançant un botó per a que l'usuari el cancel·li. En cas de prémer-lo, s'invocarà el cas d'ús "anul·lar reserva" el qual, després de demanar confirmació a l'usuari, cancel·larà la reserva.

Aquesta pantalla també permetrà anul·lar les reserves sempre que es trobin dins un marge de temps de seguretat.

5.6 Manteniment de dades mestre

5.6.1 DIAGRAMA



5.6.2 DESCRIPCIÓ

Disponible únicament pels usuaris del backoffice, es tracta del mòdul que ha de permetre el manteniment dels catàlegs necessaris pel manteniment del sistema.

Es tracta d'un mòdul genèric que ha de permetre mantenir totes les dades mestres de l'aplicació, incloent:

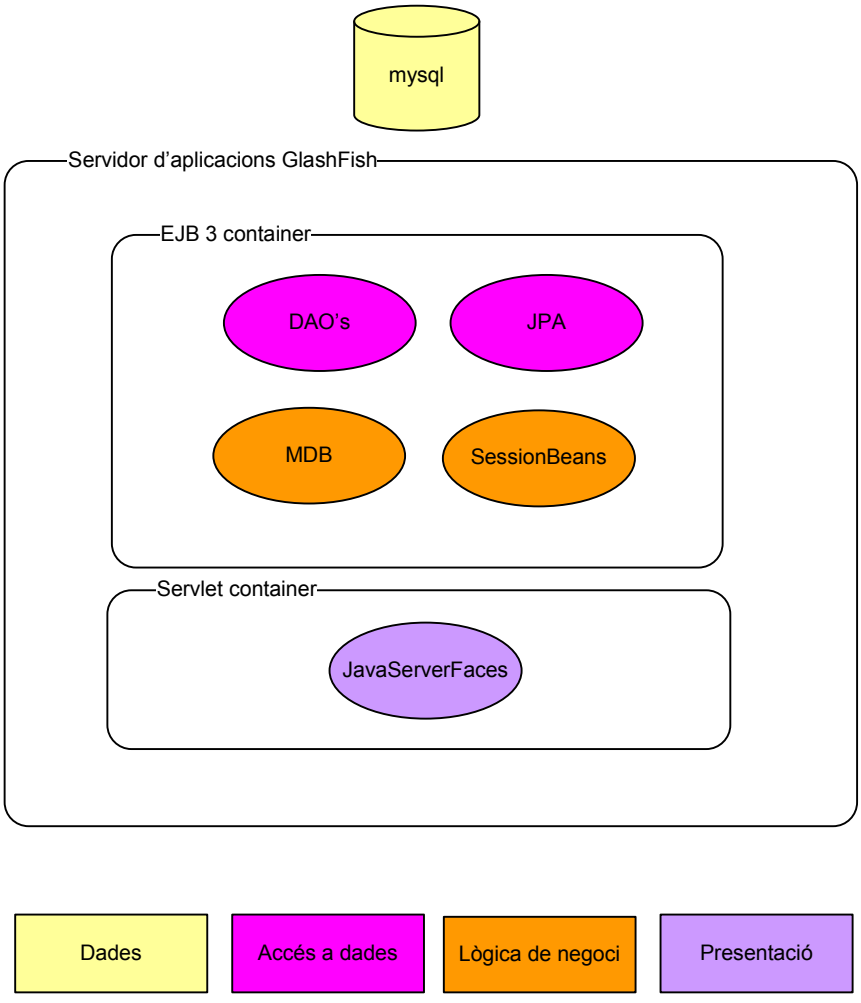
- Aeroports
- Companyies
- Vols
- Etc.

6. DISSENY TÈCNIC

6.1 Arquitectura proposada

6.1.1 DESCRIPCIÓ GENERAL

L'arquitectura proposada es descriu al següent diagrama:



Tal i com es pot observar:

- La capa dades serà implementada en una base de dades MySql

- Dins el GlashFish, la capa de presentació s'implementarà mitjançant JavaServerFaces (via RichFaces) i es desplegarà al contenidor de servlets
- La capa de lògica de negoci s'implementarà mitjançant MDB's i SessionBeans i desplegant al contenidor EJB3
- Finalment, la capa d'accés a dades s'implementarà via DAO's, JPA i Entity Beans (fent servir TopLink), sempre al contenidor EJB3

6.1.2 PATRONS APLICATS

A continuació es descriuen els principals patrons de disseny que s'empraran per a la construcció de l'aplicació.

6.1.2.1 Model Vew Controller (MVC)

El patró model-vista-controlador és un estàndard de facto dins l'arquitectura del programari consistent en separar la part del codi encarregada de la gestió de les dades (model), la corresponent a la interfície d'usuari (vista) i la lògica de negoci (controlador). El sistema implementarà aquest patró seguint el següent esquema:

- **Model:** format pel propi model (classes que representen el domini de l'aplicació.
- **Vista:** mitjançant jsf.
- **Controlador:** mitjançant els beans que respondran a les accions dels dins els jsf i una capa intermèdia mitjançant sessionbeans,), els DAO, que encapsularan l'accés a base de dades i MDB's.

6.1.2.2 FrontController

Consisteix en unificar el sistema de gestionar les peticions d'usuaris mitjançant un sol controlador, una sola classe, que s'encarregui, després, de delegar aquesta petició a diferents components. El propi framework JSF implementa aquest patró, doncs prové un servlet propi que fa de controlador i delega la gestió als diferents beans seguint la configuració del fitxer faces-config.xml.

6.1.2.3 Data Acces Objects (DAO)

El patró Data Acces Objects consisteix a encapsular els accessos al sistema de persistència mitjançant una capa de programari que ofereixi una interfície simple i unificada a la resta del sistema. Així s'aconsegueix independitzar l'aplicació del seu sistema de persistència, oferint un sistema resultant menys acoblat entre els seus components.

Aquest patró s'implementarà dins l'AirBooking mitjançant una sèrie de classes (una per cada bloc classes model) que permetrà guardar les dades del model i recuperar-les de la base de dades. Aquestes seran les úniques classes que interactuaran amb el sistema de persistència JPA.

6.2 Programació genèrica

6.2.1 DESCRIPCIÓ

Tal i com s'ha dit, un dels principals objectius del TFC és el d'aprofundir en les tecnologies emprades i veure com les seves noves característiques ajuden reduir el temps de desenvolupament i a millorar la qualitat del programari obtingut.

Una de les principals característiques de la versió 5 del JDK és la inclusió de la programació genèrica. Aquesta consisteix bàsicament en poder declarar referències a tipus de dades, classes, dins el nostre codi. D'aquesta forma podem codificar comportaments comuns per diferents classes sense tenir en compte quina és i en temps d'execució decidir el tipus.

Per exemple, per recorre una llista sense programació genèrica, necessitaríem un codi com el següent:

```
ArrayList llista = donaUnaLlistaDeLlibres();
for (Object o: llista)
{
        Llibre l = (Llibre)o;
        System.out.println(l.getTitol());
}
```

En canvi, amb programació genèrica:

```
ArrayList<Llibre> llista = donaUnaLlistaDeLlibres();
for (Llibre l: llista)
{
        System.out.println(l.getTitol());
}
```

Tot i que aquesta millora pugui semblar poc si pensem, per exemple en una de les adopcions que es fan al projecte, els DAO, els beneficis són evidents. En un sistema sense genèrics, crearíem un DAO per cada classe del nostre model (DAOLlibres, DAOAutors, etc.) on estaríem repetint sempre el mateix codi (guarda, carrega, esborra, etc.). No seria

possible crear una classe comuna ni una interfície, doncs la signatura del mètodes seria difernent:

```
public Llibre carregaLlibre();
public Autor carregaAutor();
...
```

O bé, caldria treballar amb una classe superior, per exemple Object, perdent el tipat fort, una de les principals característiques de Java.
En canvi, gràcies a la programació genèrica, es pot definir una classe principal com la següent, on T és el paràmetre genèric de tipus de classe amb la que treballem:

```
public class DAOGeneric<T>
{
        public T carrega(int id)
        {
        ....
        }
        public void guarda(T item)
        {
        ...
        }
        ...
}
```

I, després, per obtenir un DAO de Llibres, només haurem d'instanciar-lo:

```
DAOGeneric<Llibre> daoLlibres = new DAOGeneric<Llibre>();
Llibre l = daoLlibres.carrega(3); //NO hi ha cast
```

O bé, si volem afegir-li mètodes específics, heredar...

```
public class DAOLlibres extends DAOGeneric<Llibre>
{
        public List<Llibres> llistaDeLibres()
        {
                ...
        }
}
```

Dins l'àmbit del projecte, s'especifiquen tres interfícies genèriques, amb les seves corresponents implementacions per defecte.

6.2.2 EAOGENERIC

Defineix la interfície general que han de complir tots els DAO (EAO és de Entity Access Object, però és el mateix patró que el DAO).

Paràmetres genèrics:

- **T:** classe entitat que es vol persistir

Principals mètodes:

- T guarda(T elem)
- T recuperaUnItem(inti d)
- void esborra(T elem)

6.2.3 EAOPAGINABLE

Defineix la interfície que ha de complir en DAO per tal de poder ser paginable.

Paràmetres genèrics:

- **T:** classe entitat que es vol paginar

Principals mètodes:

- List <T> llistaPaginada(int min, int max)
- int numeroRegistres()

6.2.4 DATAMODELGENERIC

El component de RichFaces emprat per pintar taules de forma paginada necessita una classe en la banda del servidor que hereti de SerializabeDataModel, oferint la implementació dels mètodes que aquest necessita.

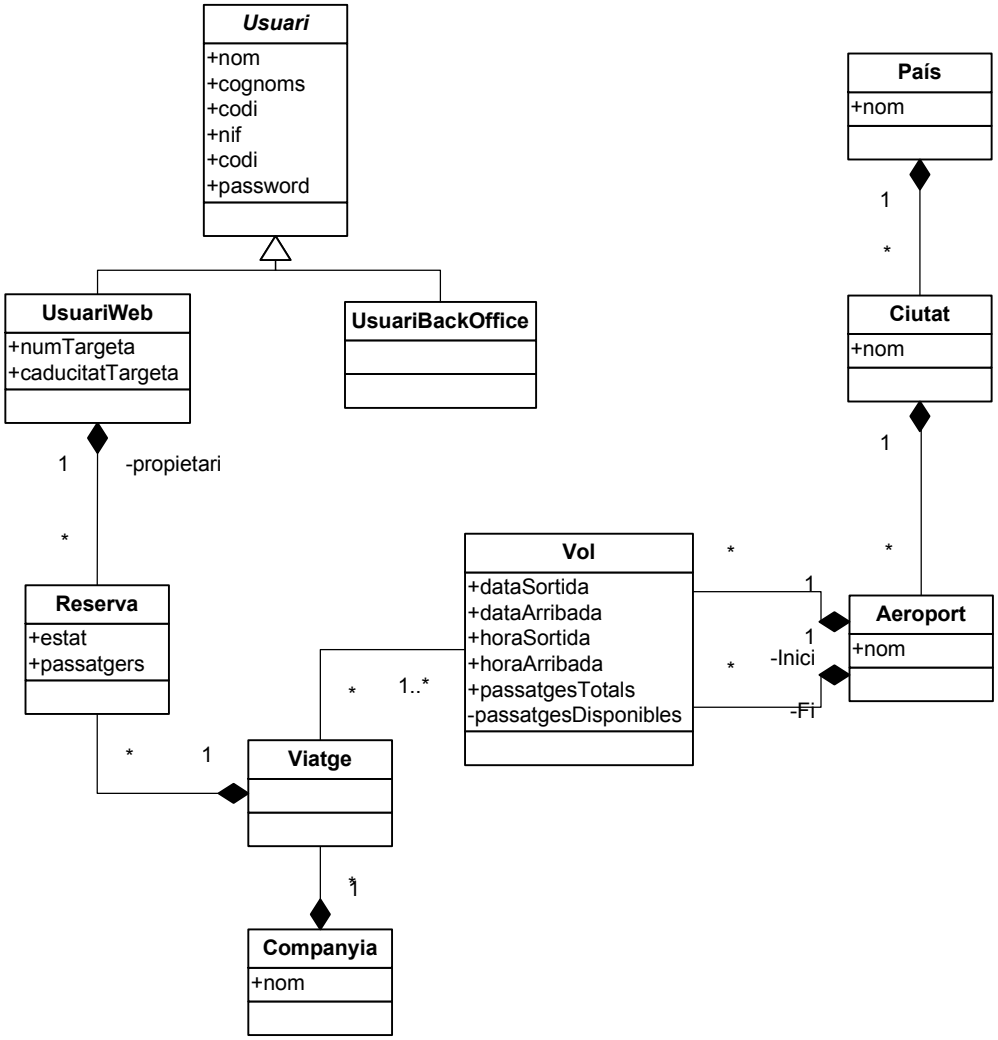
Aquesta classe farà de base per tots els datamodels del projecte. No es llisten el mètodes per ser masses i propis de la implementació e RichFaces.

Paràmetres genèrics:

- **T:** classe entitat que es vol gestionar
- **E extends EAOPaginable<T>:** EAOPaginable que oferirà l'accés a dades

6.3 Diagrama estàtic de classes

6.3.1 DIAGRAMA



6.3.2 DESCRIPCIÓ DE LES PRINCIPALS CLASSES

- **Usuari:** classe abstracta per definir tots els usuaris del sistema. Inicialment s'especificaran dues especialitzacions d'aquesta: UsuariWeb i usuariBackOffice
- **Vol:** representa un sol vol, sense escales. Es composa d'un aeroport inicial, un de destí, l'hora de sortida i la d'arribada.
- **Viatge:** representa el concepte de vol que veurà un usuari web. És a dir, conté un seguit de Vol que son les escales de les que està composat. Si es tracta d'un viatge directe, només contindrà un vol. Cal assignar-li una companyia**.**

- **Reserva:** representa la reserva efectuada per un UsuariWeb, que en serà el propietari. La reserva serà d'un sol viatge, però podrà tenir un nombre passatges definit pels disponibles pel viatge, que es calcularan a partir dels seus vols

6.4 Diagrames de seqüència

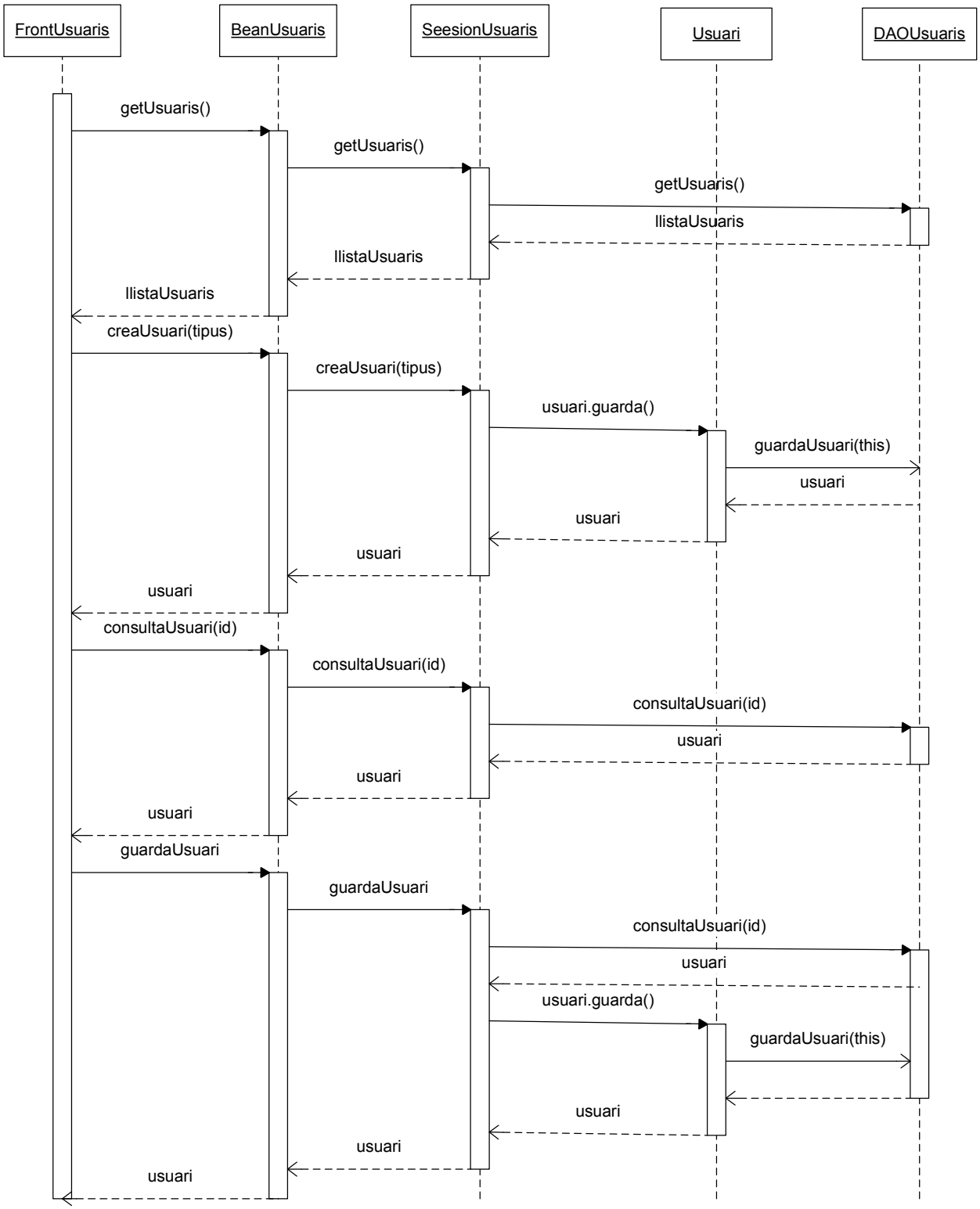
Als diagrames de seqüència es presenta la interacció entre les diferents capes de l'aplicació per tal de dur a terme les diferents tasques.

S'han especificat 5 elements d'interacció als diagrames dins les tres capes clàssiques, d'esquerra a dreta:

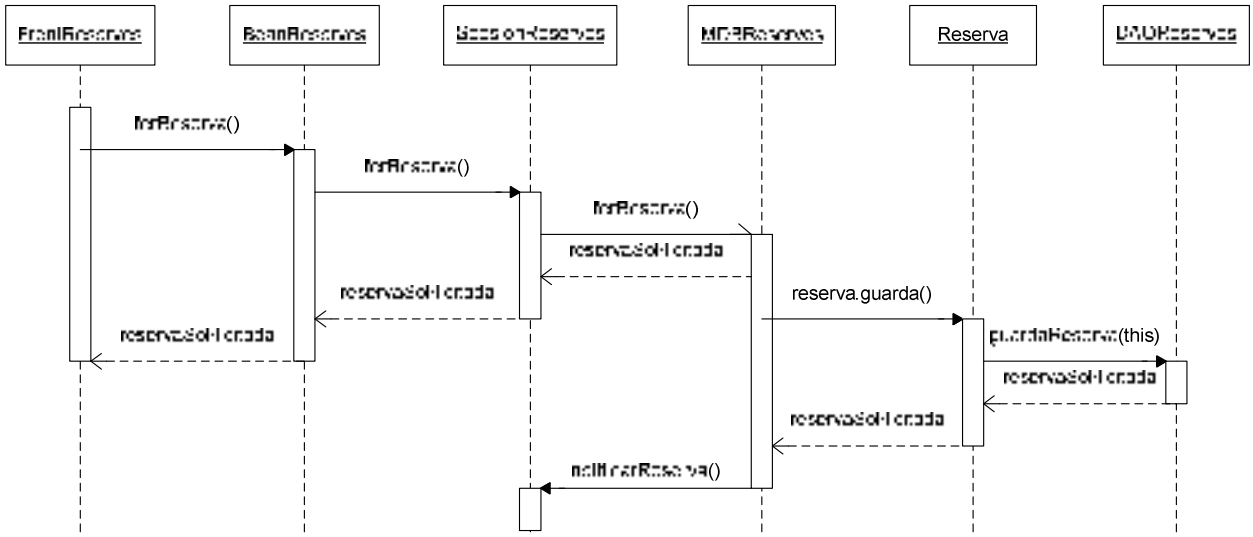
- **Presentació:**
  - **Frontera:** es tracta de les jsf i representen una acció de l'usuari
  - **Bean:** és el codi encarregat de respondre a les accions dels usuaris seguint l'estendard jsf. Són els encarregats de fer la crida a les classes de negoci
- **Lògica de negoci:**
  - **SessionBeans:** són les classes que implementen bona part de la lògica de negoci i que es desplegaran com a sessionbeans i MDB al contenidor EJB3.
  - **Model:** són el conjunt de classes que representen el domini de l'aplicació.
- **Dades**
  - **DAO:** són les classes que, mitjançant JPA, s'encarregaran de persistir i recuperar les classes del model

En els següents punts es presenten el diagrames de seqüència que s'han considerat més representatius.

6.4.1 Diagrama de seqüència de la gestió d'usuaris


FrontUsuaris
BeanUsuaris
SeesionUsuaris
Usuari
DAOUsuaris
getUsuaris()
getUsuaris()
getUsuaris()
llistaUsuaris
llistaUsuaris
llistaUsuaris
creaUsuari(tipus)
creaUsuari(tipus)
usuari.guarda()
guardaUsuari(this)
usuari
usuari
usuari
usuari
consultaUsuari(id)
consultaUsuari(id)
consultaUsuari(id)
usuari
usuari
usuari
guardaUsuari
guardaUsuari
consultaUsuari(id)
usuari
usuari.guarda()
guardaUsuari(this)
usuari
usuari
usuari

6.4.2 DIAGRAMA DE SEQÜÈNCIA DE LA RESERVA DE VOLS

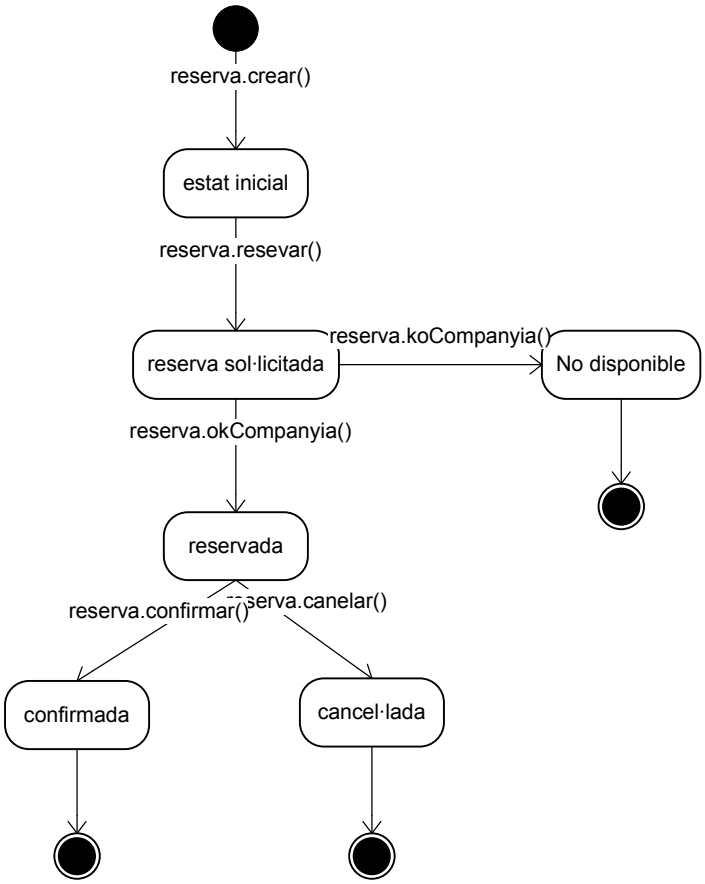


Cal destacar en aquest diagrama l'ús d'un missatge asíncron entre el *SessionReserves* i l' *MDBReserves*. L'objectiu és que l'MDB implementi la crida als serveis dels diferents sistemes que proporcioni cada companyia aèria per realitzar la reserva. En el moment que es realitza la petició, es retorna un missatge fins a la classe frontera per tal de notificar a l'usuari d'aquest fet. En canvi, quan la reserva realment s'ha efectuat, no es pot retornar fins a aquesta classe frontera, si no que serà des del propi contenidor EJB que s'enviarà un correu a l'usuari indicant-li el canvi d'estat de la reserva.

6.5 Diagrames d'estat

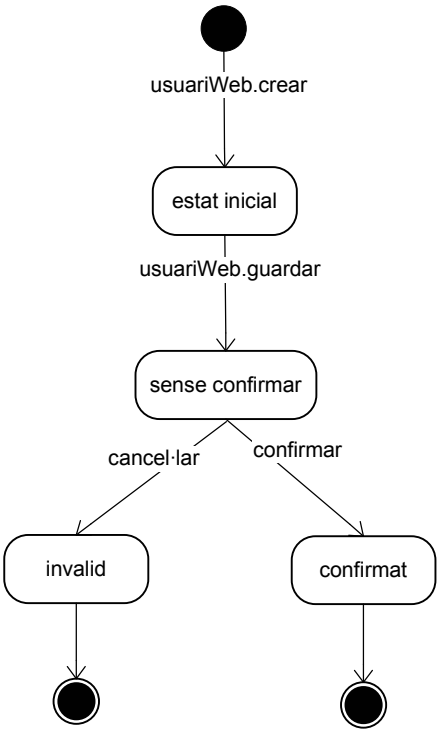
A continuació es presenten els dos diagrames d'estat que s'han considerat més representatius del sistema.

6.5.1 DIAGRAMA D'ESTATS DE LA RESERVA



En el moment de crear la reserva aquesta es crea amb estat inicial, doncs encara no hem escrit a ala base de dades. En el moment que l'usuari sol·licita la reserva, aquesta passa a estat reserva sol·licitada mentre el sistema es comunica amb el de la companyia per fer la reserva. Si la companyia no pot confirmar la reserva, aquesta passa a no disponible. Si sí pot, passa a reservada. Finalment, si l'usuari cancel·la la reserva, aquesta passa a estar cancel·lada, si no, en passar la data del vol, estarà confirmada.

6.5.2 DIAGRAMA D'ESTATS D'USUARI WEB



En el moment de crear-lo l'usuari web estarà en un estat inicial, el qual passarà a ser "sense confirmar" un cop s'hagi guardat. En aquest moment, el sistema enviarà un correu a l'usuari amb dos enllaços: un per confirmar l'alta d'usuari, que el farà passar a estat confirmat, i un per cancel·lar-lo, que passarà a estat invàlid.

7. INSTRUCCIONS D'INSTAL·LACIÓ

7.1 Base de dades

1) Instal·lar el servidor de base de dades MySql amb les opcions per defecte. La versió utilitzada ha estat la 5.0.45. Es pot obtenir de la següent url: http://dev.mysql.com/get/Downloads/MySQL-5.0/mysql-5.0.45-win32.zip/from/http://mysql.rediris.es/
2) Engegar el servei de mysql
3) Obrir una consola de mysql i executar la comanda: "create database AirBooking;"
4) Obrir una consola de ms-dos i, des del directori bin dins la carpeta on s'ha instal·lat el mysql, executar la següent comanda: "mysql -u root AirBooking < script.sql" fent que "script.sql" contingi la ruta sencera a l'script adjunt

7.2 Servidor d'aplicacions

1) Instal·lar el servidor d'aplicacions Glassfish seguint les instruccions que es poden veure al següent blog: http://weblogs.javahispano.org/lasterra/entry/instalando_glassfish
2) La versió emprada ha estat la 2, que es pot obtenir de la següent url: https://glassfish.dev.java.net/downloads/v2-b58g.html
3) Arrencar el servidor d'aplicacions i comprovar que el servidor està ben instal·lat i corrents a http://localhost:8080/
4) Aturar el servidor d'aplicacions
5) Copiar el fitxer log4j-1.2.12.jar adjunt a la carpeta lib dins el servidor
6) Copiar el fitxer mysql-connector-java-5.1.5-bin.jar dins la carpeta domains\domain1\lib dins el servidor
7) Substituir el fitxer el fitxer "domain.xml" que es troba a la carpeta "domains\domain1\config" dins la carpeta on s'ha instal·lat el glassfish pel fitxer adjunt amb el mateix nom. En reiniciar el servidor, apareixeran el "JDBC Resource" i el "Connection Pool" corresponent. Cal validar-ho des de la consola d'administració.
    a. En cas de produir-se algun error, la configuració manual és la següent:
        - Coonection Pool:
            - Name: AirBookingPool
            - Datasource Classname: com.mysql.jdbc.jdbc2.optional.MysqlDataSource
            - Resource Type: javax.sql.DataSource
            - A "Additional Pool Properties":

        a. <property name="databaseName" value="AirBooking"/>
        b. <property name="serverName" value="localhost"/>
        c. <property name="password" value="AirBooking"/>
        d. <property name="user" value="AirBooking"/>
    - La resta amb valors per defecte
- DataSource:
    - JNDI Name: jdbc/AirBooking
    - PoolName: AirBookingPool
- Cua JMS:
    - JNDI Name: jms/AirBookingQueue
    - Res-type: javax.jms.Queue
    - <property name="Name" value="AirBookingQueue"/>
- Connetion Factory:
    - JNDI Name: jms/AirBookingConnectionFactory
    - La resta d'atributs per defecte.

8) A la consola d'administració, anar a la opció Applications -> Enterprise Applications, fer clic a "deploy" i seleccionar el fitxer AirBookingEAR.ear adjunt, amb totes les opcions en blanc
9) Accedir a l'aplicació mitjançant la URL: http://localhost:8080/AirBooking
10) Inicialment existeixen dos usuaris:
    a. Per accedir a la part d'administració: Administrador / Administrador
    b. Per accedir a la part d'usuari: Usuari / Usuari

7.3 Entorn de desenvolupament

Dins la carpeta "workspace" adjunta hi ha les tres carpetes dels tres projectes (la wepapp, l'ejb i el ear), que copiant-les a la carpeta arrel d'un workspace hauria de veure's bé. El projecte s'ha desenvolupat amb eclipse IDE 3.3 (Europa) i el plug-in del glassfish es pot obtenir a https://glassfishplugins.dev.java.net/

8. BIBLIOGRAFIA

GEARY, D; HORSTMANN, C. *Core JavaServer Faces*. Sun Microsystems. http://horstmann.com/corejsf

PANDA, D; REZA, R; LANE, D. *EJB 3 in action*. Manning. http://www.manning.com/panda/

RichFaces Developer Guide. Red Hat