NANYANG TECHNOLOGICAL UNIVERSITY

Individual Project-2
Airline Booking System

FUNCTIONAL SPECIFICATION

COURSE TITLE	**:** CI6225 – ENTERPRISE APPLICATION DEVELOPMENT
LECTURER	**:** Mr. HARIHARAN VAIDHYANATHAN
STUDENT	**:** Adam Myrén

1 Background of the system

The Airline Booking System developed is to be used as a tool for airlines that which want to handle their customer interactions, such as bookings and customer registrations, and manage their routes via a web interface. It provides admin functionally, such as being able to retrieve a list of all customers. Except being able to book new flights, customers need to be able to view and manage all their previous bookings. Airline personnel needs to be able to add new flights and modify existing flights.

2 System Scope

2.1 Functional Requirements

Based on the background presented functional requirements has been developed in order to specify what functions that are required from the system. Due to time constraints, all requirements are not included in the first deliverable. The requirements that has been implemented are: Registration of user, login of user, search for flights, book a flight, view booked flights, view booked flights, show report of passengers on a given flight, show report of customers, modify a route and add a new route as these were considered to be part of the core functionality. In the table below, all functional requirements of the system are stated.

Functional Requirements
Registration of user
Login of user
Search for flights
Book a flight
View booked flights
Cancel flights
Show report of passenger on a given flight
Show report of all customers
Show financial report for a period of time
Modify a booking
Modify a route
Add a new route
Create new admins
Check in on a flight
Delete user
Add Promotion on flight route
Delete Promotion on flight route
Display promotion routes for customers visiting the page

2.2 Users

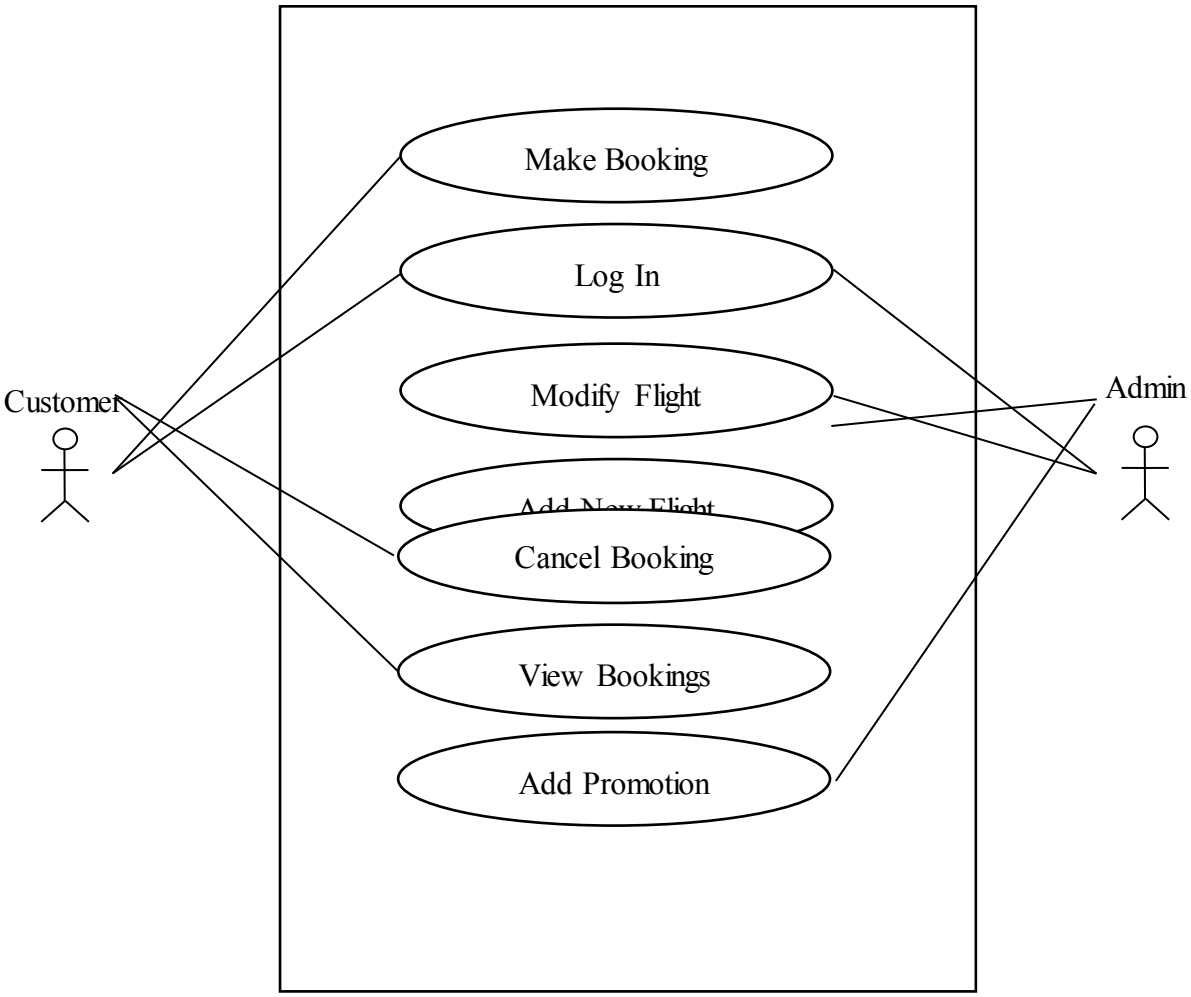
Both administrators from the airline company and the customers of the airline company should be able to use the system and its functionalities.

2.3 Assumptions

3 Analysis of the system

3.1 Use case diagram

Below is a use case diagram based on some of the requirements of the application.



3.2 Database analysis

For persistence handling a relational database has been used. An entity relationship diagram has been developed and later translated into relational database tables. There are five tables in the database: Airports, Bookings, Flights, Passengers, and Users. They all have one primary key, except for flight that is uniquely defined by an id and a date. A booking is a week entity, that is uniquely defined by the flight it regards and the passenger that the passenger that that is registered on the booking. On top of these entities, three additional view have been created, PromotionView (origin Airport destination Airport, lowest price), MyBookingsElement (flightId, date, departureTime, duration, origin, destination, firstname, lastname, passportNr, user) and AdminPassengerView(flightId, date, passportNr, firstname, lastname, passportExp, passportIsh, dateOfBirth).

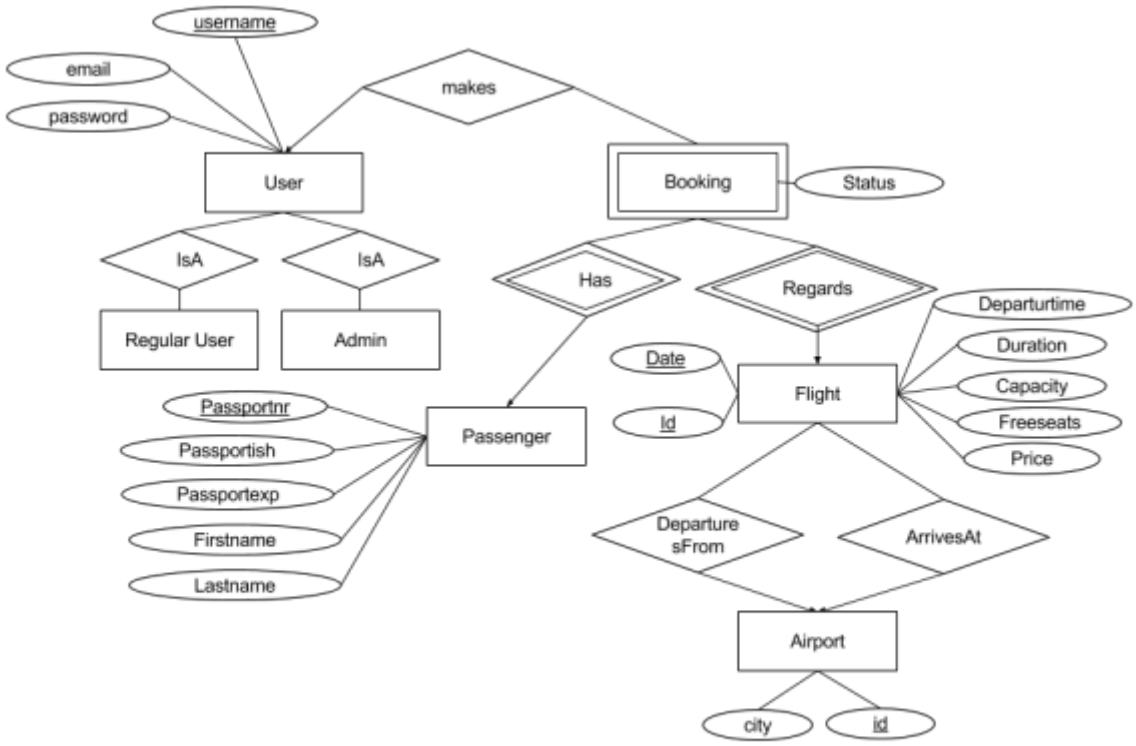


Figure 1. Database architecture

4 System Design

4.1 Class diagram

The system applies the Struts2 framework and uses Hibernate for persistence. One interceptor is user (LoginInterceptor) that checks if a user is logged in when a blocked page is loaded. If a user is not logged in, the interceptor will instead redirect the user to a login page. All of the action classes has an execute method that is being called when a request is being made. The business service layer has been added so that it will be easier to scale the application in the future. It allows many different action classes to use the same methods which improves reusability of the code. The business service methods communicate with the persistence layer through an interface that hides the implementation from the business logic. In this application HibernateDAO is the implementation of the interface and all implementations of the DAO extends an abstract class, AbstractHibernateDAO, that has basic methods for getting, adding and deleting entities. All entities extend the abstract class AbstractEntity, which makes the implementation of AbstractHiberneteDAO possible. There is also a Utility class, providing methods mainly used to validate input. The JSP pages to not use scriptlets in order to minimize the amount of logic carried out in the view, instead struts2 tags are being used.

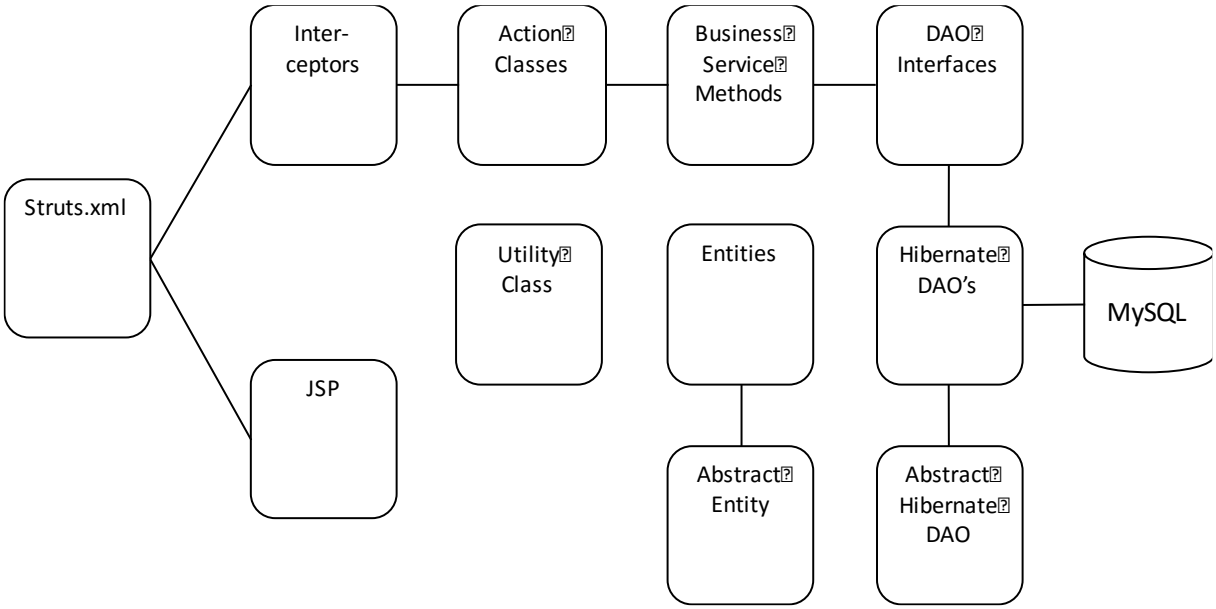


Figure 2. System Architecture

4.2 Entity Classes

The entity classes are either representations of tables or views in the mysql database. Their objective is to collect data related to one object so that it easily can be transferred between other classes. When a composite key is used in the relational database a new class is created for that key, in that case the naming convention is <name of entity>Id.

4.2.1 Airport

Representation of the airports table in the database with getters and setters for all attributes.

Class Name	Airport		
Type	Plain Java Class		
Package	entities		
Constructor	Airport(String id, String city)		
Attribute	airportId	String	ID of the airport
	airportCity	String	City of the airport
Method	getAirportId(): String		Getter method for id
	setAirportId(String id): void		Setter method for id
	getAirportCity(): String		Getter method for city
	setAirportCity(String city): void		Setter method for city

4.2.2 Booking

Representation of the bookings table in the database with getters and setters for all attributes.

Class Name	Booking		
Type	Plain Java Class		
Package	entities		
Constructor	Booking(BookingId id, String user, String status)		
Attribute	id	BookingId	Flight that the booking regards
	user	String	User that has created the booking
	status	String	Status of the booking
Method	getId(): BookingId		Getter method for id
	setId(BookingId id): void		Setter method for id
	getUser():String		Getter method for user
	setUser(String user):void		Setter method for user
	getStatus():String		Getter method for status
	setStatus(String status):void		Setter method for status
	toString():String		Override toString

4.2.3 BookingId

The id class for the Booking Entity.

Class Name	BookingId		
Type	Plain Java Class		
Package	entities		
Constructor	BookingId(String flight_id, String passenger, String date)		
Attribute	flightId	String	Flight that the booking regards
	passenger	String	Passenger that the booking regards
	date	String	Date that the booking concerns
Method	getFlightId(): String		Getter method for id

	setFlightId(String flightId): void	Setter method for id
	getPassenger(): String	Getter method for city
	setPassenger(String passenger): void	Setter method for city
	getDate():String	Getter method for date
	setDate(String date):void	Setter method for date
	toString():String	Override toString

4.2.4 Flight

Representation of the airports table in the database with getters and setters for all attributes.

Class Name	**Flight**		
Type	**Plain Java Class**		
Package	**entities**		
Constructor	Flight(FlightId id, String departure_time, String duration, int capacity, String origin, String destination)		
Attribute	id	flightId	ID of the flight
	departureTime	String	Departure time of flight
	duration	String	Duration of the flight
	capacity	int	Capacity of the aircraft
	freeSeats	int	Number of free seats left on the flight
	origin	String	ID of origin airport
	destination	String	ID of destination airport
	price	String	Ticket price for the flight
	promotion	String	States if the flight has a promotion
Method	getId():FlightId		Getter method for id
	setId(FlightId): void		Setter method for id
	getDeparture_time():String		Getter method for departure_time
	setDeparture_time(String departure_time): void		Setter method for departure_time
	getDuration():String		Getter method for duration
	setDuration(String duration):void		Setter method for duration
	getCapacity(): int		Getter method for capacity
	setCapacity(int capacity):void		Setter method for capacity
	getFreeSeats(): int		Getter method for freeSeats
	setFreeSeats(int freeSeats):void		Setter method for freeSeats
	getOrigin():String		Getter method for origin
	setOrigin(String origin):void		Setter method for origin
	getDestination():String		Getter method for destination
	setDestination(String destination):void		Setter method for destination
	getPrice():String		Getter method for price

	setPrice(String price):void	Setter method for price
	toString():String	Override toString

4.2.5 FlightId

The id class for the Flight entity.

Class Name	FlightId		
Type	Plain Java Class		
Package	entities		
Constructor	FlightId(String id, String date)		
Attribute	flightId	String	ID of the flight
	date	String	Date of the flight
Method	geFlightId():String		Getter method for id
	setFlightId(String _id): void		Setter method for id
	getDate():String		Getter method for date
	setDate(String date):void		Setter method for date
	toString():String		Override toString

4.2.6 MyPagesBookingElement

A representation of a join of two tables in the database, used to display information regarding bookings to the user.

Class Name	MyPagesBookingElement		
Type	Plain Java Class		
Package	entities		
Constructor	MyPagesBookingElement(String id, String date, String departure_time, String duration, String origin, String destination, String firstname, String lastname, String passportnr)		
Attribute	id	String	ID of the flight
Method	getId():MyPagesBookingElementId		Getter method for id
	setId(MyPagesBookingElementId id): void		Setter method for id
	toString():String		Override toString

4.2.7 MyPagesBookingElementId

The id class for booking element.

Class Name	MyPagesBookingElementId		
Type	Plain Java Class		
Package	entities		
Constructor	MyPagesBookingElementId(String flightId, String date, String departure_time, String duration, String origin, String destination, String firstname, String lastname, String passportnr)		
Attribute	flightId	String	ID of the flight
	date	String	Date of the flight
	departureTime	String	Departure time of flight
	duration	String	Duration of the flight

	origin	String	ID of origin airport
	destination	String	ID of destination airport
	firstname	String	First name of passenger
	lastname	String	Last name of passenger
	passportNr	String	Passport number of passenger
	user	String	User associated with the element
Method	getId():String		Getter method for id
	setId(String _id): void		Setter method for id
	getDate():String		Getter method for date
	setDate(String date):void		Setter method for date
	getDeparture_time():String		Getter method for departure_time
	setDeparture_time(String departure_time): void		Setter method for departure_time
	getDuration():String		Getter method for duration
	setDuration(String duration):void		Setter method for duration
	getCapacity(): int		Getter method for capacity
	setCapacity(int capacity):void		Setter method for capacity
	getOrigin():String		Getter method for origin
	setOrigin(String origin):void		Setter method for origin
	getDestination():String		Getter method for destination
	setDestination(String destination):void		Setter method for destination
	getFirstname():String		Getter method for firstname
	setFirstname(String firstname):void		Setter method for firstname
	setLastname():String		Getter method for lastname
	setLastname(String lastname):void		Setter method for lastname
	getPassportnr():String		Getter method for passportnr
	setPassportnr(String passportnr):void		Setter method for passportnr
	getUser():String		Getter method for user
	setUser(String user):void		Setter method for user
	toString():String		Override toString

4.2.8 Passenger

Representation of the passengers table in the database with getters and setters for all attributes.

Class Name	**Passenger**		
Type	**Plain Java Class**		
Package	**entities**		
Constructor	Passenger(String firstname, String lastname, String passportNr, String passportExp, String passportIsh, String dateOfBirth)		
Attribute	firstname	String	First name of passenger
	lastname	String	Last name of passenger

	passportNr	String	Passport number of passenger
	passportExp	String	Expiry date of passport
	passportIsh	String	Date of issue of passport
	dateOfBirth	String	Date of birth
Method	getFirstname():String		Getter method for firstname
	setFirstname(String firstname):void		Setter method for firstname
	getLastname():String		Getter method for lastname
	setLastname(String lastname):void		Setter method for lastname
	getPassportNr():String		Getter method for passportNr
	setPassportNr(String passportnr):void		Setter method for passportNr
	getPassportExp():String		Getter method for passportExp
	setPassportExp(String passportexp):void		Setter method for passportExp
	getPassportIsh():String		Getter method for passportIsh
	setPassportIsh(String passportish):void		Setter method for passportIsh
	getDateOfBirth():String		Getter method for dateOfBirth
	setDateOfBirth(String birth):void		Setter method for dateOfBirth
	toString():String		Override toString

4.2.9 AdminPassengerView

A view created to display passengers booked on a specific flight on the admin page.

Class Name	**AdminPassengerView**		
Type	**Plain Java Class**		
Package	**entities**		
Constructor	AdminPassengerView(AdminPassengerViewId id)		
Attribute	id	AdminPassengerViewId	Id of the passengerview element
Method	getId():AdminPassengerViewId		Getter method for id
	setId(AdminPassengerViewId id):void		Setter method for id
	toString():String		Override toString

4.2.10 AdminPassengerViewId

Id class for AdminPassengerView.

Class Name	**AdminPassengerViewId**
Type	**Plain Java Class**
Package	**entities**
Constructor	AdminPassengerViewId(String flightId, String date, String firstname, String lastname, String passportNr, String passportExp, String

	passportIsh, String dateOfBirth)		
Attribute	flightId	String	Id of flight
	date	String	Date of flight
	firstname	String	First name of passenger
	lastname	String	Last name of passenger
	passportNr	String	Passport number of passenger
	passportExp	String	Expiry date of passport
	passportIsh	String	Date of issue of passport
	dateOfBirth	String	Date of birth
Method	getFlightId():String		Getter method for flightId
	setFlightId(String flightId):void		Setter method for flightId
	getDate ():String		Getter method for date
	setDate(String date):void		Setter method for date
	getFirstname():String		Getter method for firstname
	setFirstname(String firstname):void		Setter method for firstname
	getLastname():String		Getter method for lastname
	setLastname(String lastname):void		Setter method for lastname
	getPassportNr():String		Getter method for passportNr
	setPassportNr(String passportnr):void		Setter method for passportNr
	getPassportExp():String		Getter method for passportExp
	setPassportExp(String passportexp):void		Setter method for passportExp
	getPassportIsh():String		Getter method for passportIsh
	setPassportIsh(String passportish):void		Setter method for passportIsh
	getDateOfBirth():String		Getter method for dateOfBirth
	setDateOfBirth(String birth):void		Setter method for dateOfBirth

4.2.11 User

Representation of the users table in the database with getters and setters for all attributes.

Class Name	**User**		
Type	**Plain Java Class**		
Package	**entities**		
Constructor	User(String username, String password, String email, String admin)		
Attribute	username	String	Username of user
	password	String	Password of user
	email	String	Email of user
	admin	String	Indicates whether user has admin privileges
Method	getUsername():String		Getter method for username

	setUsername(String username):void	Setter method for username
	getPassword():String	Getter method for password
	setPassword(String password):void	Setter method for password
	getEmail():String	Getter method for email
	setEmail(String email):void	Setter method for email
	getAdmin():String	Getter method for admin
	setAdmin(String admin):void	Setter method for admin

4.2.12 Promotion

A promotion is a view in the relational database, created to keep track of all routes that are currently promoted, but also the lowest available price on each route.

Class Name	**Promotion**		
Type	**Plain Java Class**		
Package	**entities**		
Constructor	Promotion(PromotionId id)		
Attribute	id	PromotionId	ID of the promotion
Method	getId(): PromotionId		Getter method for id
	setId(PromotionId id): void		Setter method for id
	toString():String		Override toString

4.2.13 PromotionId

The id class for promotion.

Class Name	**PromotionId**		
Type	**Plain Java Class**		
Package	**entities**		
Constructor	PromotionId(String originCity, String originId, String destinationCity, String destinationId, String price)		
Attribute	originId	String	ID of the origin airport
	originCity	String	City of the origin airport
	destinationId	String	ID of the destination airport
	destinationCity	String	City of the destination airport
	price	String	Lowest available price on route
Method	getOriginId(): String		Getter method for origin id
	setOriginId(String originId): void		Setter method for origin id
	getOriginCity(): String		Getter method for origin city
	setOriginCity(String originCity): void		Setter method for origin city
	getDestinationId(): String		Getter method for destination id
	setDestinationId(String destinationId): void		Setter method for destination id
	getDestionationCity(): String		Getter method for destination city

	setDestinationCity(String destinationCity): void	Setter method for destination city
	getPrice(): String	Getter method for price
	setPrice(String price): void	Setter method for price

4.3 Data Access Objects

The data access objects exist to separate the business layer from the database implementation, there is one DAO for each table in the database. Displayed below are all the interfaces for the dao's, one for each entity. These interfaces are then implemented using Hibernate.

4.3.1 AirportDAO

Data access object handling all requests towards the Airports table in the database.

Interface Name	**AirportDAO**	
Type	**Interface**	
Package	**dao**	
Method	getAirportId(String city):List<Airport>	Return all airport id's for a given city
	Add(Airport airport):boolean	Add a new airport
	Exists(Airport Airport):boolean	Check if an airport exists
	Get(AirportId id): Airport	Get a specific airport
	getAll(String city):List<Airport>	Get all airports

4.3.2 BookingDAO

Data access object handling all requests towards the Bookings table in the database.

Interface Name	**BookingDAO**	
Type	**Interface**	
Package	**dao**	
Method	Add(Booking booking):boolean	Adds a new booking to the database
	getBookings(String username):List<Booking>	Returns a list of all bookings related to a given user
	getBookings(Flight flightIdList<Booking>	Returns a list of all bookings on a given flight
	Delete(BookingId bookingId, Class<booking> theClass)	Deletes a given booking
	Add(List<Booking> listOfBookings)	Adds a list of bookings in one transaction

4.3.3 FlightDAO

Data access object handling all requests towards the Flights table in the database.

Interface Name	**FlightDAO**
Type	**Interface**
Package	**dao**

Method	getFlights(String origin, String destination, String date, int nr_of_tickets):List<Flight>	Returns a list of flights between two airports on a given day that has more or equal free seats as nr_of_tickets
	get(FlightId flightId, Class<Flight> theClass):Flight	Returns a spcific flight
	exists(flightId flightId):boolean	Returns true if the flight exists, else false
	add(Flight flight):boolean	Adds a flight to the database
	Update(FlightId flightId, Flight newFlight):boolean	Updates an existing flight with new data
	Update(String originId, String destinationId, String promotion)	Updates the promotion value on all flights on a certain route

4.3.4 PassengerDAO

Data access object handling all requests towards the Passengers table in the database.

Interface Name	**PassengerDAO**	
Type	**Interface**	
Package	**dao**	
Method	get(String passportNr):Passenger	Returns a passenger based on the primary key
	add (Passenger passenger):boolean	Adds a new passenger to the database
	exists(String passportNr):boolean	Returns true if the passenger exists, else false
	getAll(Class<Passenger> theClass)	Returns all passengers.

4.3.5 AdminPassengerViewDAO

Interface Name	**AdminPassengerViewDAO**	
Type	**Interface**	
Package	**dao**	
Method	getPassengers(FlightId flightId):List<AdminPassengerView>	Returns a list of passengers on a specified flight.

4.3.6 MyBookingElementsDAO

Interface Name	**MyBookingElementsDAO**	
Type	**Interface**	
Package	**dao**	
Method	getElements(String usernameList<MyBookingsElement>	Returns all elements for a user.

4.3.7 PromotionDAO

Interface Name	**PromtionDAO**

Type	Interface	
Package	dao	
Method	getAll(Class<Promotion> theClass):List<Promotion>	Returns all promotions.

4.3.8 UserDAO

Data access object handling all requests towards the Users table in the database.

Interface Name	UserDAO	
Type	Interface	
Package	dao	
Method	get(String username, Class<User> theClass):User	Returns a user based on the primary key
	add(User user):boolean	Adds a new user to the database
	exists(String username, String password):boolean	Returns true if the user with the specified password exists, else false
	Exists(String username, Class<User> theClass)	Returns true of the user exists, else false
	getAll(Class<User> theClass):List<User>	Returns a list of all users

4.4 Business Logic Beans

These java enterprise beans handle all business logic in the system. The collect and edit persistent data through the use of DAO's and they are invoked by the different servlets.

4.4.1 AdminService

Handles all business logic related to a user with admin privileges.

Class Name	AdminService		
Type	Plain Java Class		
Package	businessService		
Constructor	AdminService()		
Attribute	flightdao	FlightDAO	DAO to handle flights
	adminPassengerViewDAO	adminPassengerViewDAO	DAO to handle adminview elements for passengers
	promotionDAO	PromotionDAO	DAO to handle promotions
	userdao	UserDAO	DAO to handle users
Method	getPassengerList(FlightId flightId):List<AdminPassengerView>		Returns a list of all passengers on a specific flight, and includes the flight information
	Add(Flight flight):boolean		Adds a new flight to the database
	getAllUsers():List<User>		Returns all users in the system
	getPromotions():List<Promotion>		Returns all promotions in the system

	setPromotion(String originId, String destinationId, String promotion)	Adds or removes a promotion on a route, depending on the value of promotion

4.4.2 BookingService

Handles all business logic involved when making a booking, from search of flights to registration of the booking.

Class Name	**BookingService**		
Type	**Plain Java Class**		
Package	**businessService**		
Constructor	BookingService()		
Attribute	flightdao	FlightDAO	DAO to handle flights
	passengerdao	PassengerDAO	DAO to handle passengers
	bookingdao	BookingDAO	DAO to handle bookings
Method	getFlights(String origin, String destination, String date, int nrOfTickets):List<Flight>		Returns a list of flights between two airports on a given day that has more or equal free seats as nrOfTickets
	bookFlights(List<Passenger> passengers, List<Flight> flights, User user):boolean		Books all passengers on all flights and associates them with the given user
	deleteBooking(BookingId bookingId):boolean		Deletes a given booking

4.4.3 UserService

Handles all type of user administration, such as adding users and administrating information related to a specific user.

Class Name	**UserService**		
Type	**Plan Java Class**		
Package	**businessService**		
Constructor	UserService()		
Attribute	userdao	UserDAO	DAO to handle users
	elementsDAO	MyBookingElementsDAO	DAO to handle bookingElements
Method	validateUser(User user):boolean		Check if a user with that username and password exists
	registerUser(User user):boolean		Adds the user to the database
	exists(String username):boolean		Checks if the user already exists
	setSession(Map<String,Object>sessionMap):void		Mandatory method for accessing the session
	getUser(String username): User		Returns a user based on the key
	getElements(User user):List<MyBookingElements>		Returns a list of booking elements associated with a user

4.5 Interceptors

Only one interceptor is used in this project. The LoginInterceptor intercepts any attempt to access a restriced page (such as MyPages or AdminPage), and verifies that a user is logged in. If a user is not logged in, then the user is redirected to the login page before proceeding to the originally intend page.

Class Name	**LoginInterceptor**	
Type	**Plan Java Class**	
Package	**businessService**	
Method	intercept(ActionInvocation actionInvocation):String	This method is invoked whenever the interceptor is called. It verifies that a user is logged in.

4.6 Action Classes

16 action classes have been developed to handle request from the view and communicate with the business logic. All action classes have one two methods, execute() (all classes) and validate() (where validation of input is required).

ActionClass Name	Functionality
AddFlightAction	Adds a new flight to the system
BookFlightAction	Books passengers on a given flight
CancelBookingAction	Deletes a booking from the database
ChangePromotionAction	Adds or removes a promotion on a specific route
CreateReportAction	Generates different types of administrator reports, depending on input
DownloadXMLAction	Creates and downloads an XML representaton of a report
ExitBookingAction	Exits a booking session and returns the user to StartPage/MyPages
GetPromotionsAction	Downloads all promotions and saves them to the valuestack
IndexAction	Redirects to the right start jsp, StartPage and MyPages respectively, depending on if a user is logged in or not
LoginAction	Aciton for logging a user in
LogoutAction	Invalidates the current session and sends the user to the StartPage
PrepareMyPagesAction	Loads necessary information (booked flights) to start MyPages.jsp
RegisterUserAction	Registers a new user to the system
savePassengerInformationAction	Saves passenger information input to the session so that it can be accessed later on in the booking process
SearchFlightAction	Gets a list of flights on a given date between two given airports and saves it to the session

SelectFlightAction	Interpreters what flight the user has picked and then attaches those two flights to the session so that they can be accessed later on in the booking process.

4.7 JSP Files

A number of JSP page have been used to display information to the user and to collect various inputs.

Name	Functionality
AdminPage.jsp	StartPage for admin users
ConfirmBooking.jsp	Page where the user can review the booking details and enter payment information
ViewReport.jsp	A page to display various reports generated by an admin user
Login.jsp	Page where one can login or create a new user
MyPages.jsp	StartPage for a logged in user
PutInformation.jsp	Page where a user that is about to make a booking can enter passenger information
SelectFlight.jsp	Page where a user that is about to make a booking can select among available flights
StartPage	Start page if no one is logged in

4.8 Supporting Files

Besides files that support the main functionalities of the system, a number of other files are needed to style the jsp pages and make them interactive for the user. Therefore, css and Javascript have been used to achieve this.

Type	Name	Description
CSS	main.css	Overall style
	login.css	Style Login.jsp
	adminpage.css	Style AdmiPage.jsp
	confirmbooking.css	Style ConfirmAndPay.jsp
	viewreport.css	Style DisplayReport.jsp
	mypages.css	Style MyPages.jsp
	putinformation.css	Style PutInformation.jsp
	selectflight.css	Style SelectFlight.jsp
Javascript	script.js	Client side script